Public Verification for Private Hash Matching

Sarah Scheffler
Princeton University
sscheff@princeton.edu

Anunay Kulshrestha
Princeton University
anunay@princeton.edu

Jonathan Mayer
Princeton University
jonathan.mayer@princeton.edu

***Abstract*—End-to-end encryption (E2EE) prevents online services from accessing user content. This important security property is also an obstacle for content moderation methods that involve content analysis. The tension between E2EE and efforts to combat child sexual abuse material (CSAM) has become a global flashpoint in encryption policy, because the predominant method of detecting harmful content—server-side perceptual hash matching on plaintext images—is unavailable.**

Recent applied cryptography advances enable private hash matching (PHM), where a service can match user content against a set of known CSAM images without revealing the hash set to users or nonmatching content to the service. These designs, especially a 2021 proposal for identifying CSAM in Apple's iCloud Photos service, have attracted widespread criticism for creating risks to security, privacy, and free expression.

In this work, we aim to advance scholarship and dialogue about PHM by contributing new cryptographic methods for system verification by the general public. We begin with motivation, describing the rationale for PHM to detect CSAM and the serious societal and technical issues with its deployment. Verification could partially address shortcomings of PHM, and we systematize critiques into two areas for auditing: *trust in the hash set* and *trust in the implementation*. We explain how, while these two issues cannot be fully resolved by technology alone, there are possible cryptographic trust improvements.

The central contributions of this paper are novel cryptographic protocols that enable three types of public verification for PHM systems: (1) certification that external groups approve the hash set, (2) proof that particular lawful content is not in the hash set, and (3) eventual notification to users of false positive matches. The protocols that we describe are practical, efficient, and compatible with existing PHM constructions.

1. Introduction

End-to-end encryption (E2EE) is an invaluable safeguard for online communications: if a service cannot access content, then a compromise of the service will not cause unauthorized disclosure of content. Governments, businesses, journalists, researchers, activists, and users worldwide depend on E2EE protections. Popular messaging services, including Signal [1], Meta's WhatsApp [2], Apple's iMessage [3], and Google's Messages [4] now offer user-friendly implementations of E2EE that are enabled by default.

But the same properties that make E2EE secure create unprecedented challenges for content moderation: If a service provider cannot access content, it cannot proactively detect and mitigate online harms on the basis of content.

This limitation has led to worldwide controversy about how E2EE interacts with efforts to combat the proliferation of child sexual abuse material (CSAM) online. The predominant method for proactively identifying CSAM is hash matching, typically by using a *perceptual hash function* (PHF) to map visually similar media to identical or similar hash values [5]–[7]. The use of hash matching for content moderation is not exclusive to identifying CSAM—major platforms currently rely on hash matching for detecting terrorist content [8], [9], intellectual property infringement [10], and nonconsensual intimate imagery [11], among other categories of problematic content. Services similarly rely on exact and approximate hash matching to detect malware executables [12] and other online security threats.

Hash matching, as presently deployed, depends on server-side analysis of plaintext content: the client sends their content to the server, which computes a perceptual hash value. The server then matches the hash value against a confidential set of hashes for known disallowed content [6]–[8], [10], [13]. This approach is not immediately compatible with E2EE because, by design, the server cannot access the client's content to compute a perceptual hash value.

Recent advances in applied cryptography enable *private hash matching* (PHM), where a service can identify a match between a client's hash and a server's hash set while learning nothing about nonmatching content and maintaining the confidentiality of the hash set. In 2021, Kulshrestha and Mayer [14] proposed constructions for exact and approximate PHM based on homomorphic encryption. Shortly after, the Apple team of Bhowmick et al. [15] proposed a construction for exact matching with thresholded content disclosure based on more efficient elliptic curve encryption, which Apple intended to deploy in its iCloud Photos service [16].

These innovations in PHM have generated immense interest from governments and child safety groups worldwide as a possible approach for CSAM detection in E2EE environments. Before Apple's 2021 announcement, officials from Australia, Canada, Japan, New Zealand, the U.K., and the U.S., among other countries, were already advocating for development of new CSAM detection methods in E2EE [17]. After the announcement, child safety groups

worldwide praised Apple's proposal [18]. The European Commission and the U.K. government also launched funding initiatives for systems to detect CSAM under E2EE [19].

Meanwhile, a number of security researchers, civil rights groups, and policymakers responded to Apple's announcement with sharp criticism, highlighting how a PHM system could undermine security, privacy, and free expression in the U.S. and abroad (e.g., [14], [20]–[30]). The concerns, which we elaborate on in Section 2.4, included pressure to expand the detection system to other kinds of content, lack of transparency, auditability, and enforcement of Apple's promises, and false positive matches. Apple responded by pausing deployment of its protocol while proceeding with other less-controversial child safety mechanisms. In December 2022, Apple announced that it had permanently suspended its plans to deploy the PHM system [31].

This work aims to advance scholarship and dialogue about PHM systems by identifying particular areas of concern that are amenable to technical transparency improvements, then constructing and implementing cryptographic protocols for those improvements. We focus on PHM systems for CSAM detection because they are the current focal point for implementation and policy discourse; much of our analysis generalizes to other methods for detecting content and other categories of harmful content.

We begin in Section 2 with background on online CSAM detection. We discuss the risks and costs of implementing detection systems under E2EE, and we systematize critiques of PHM into two areas: trust in the hash set used for matching and trust in the implementation of the system. These areas are amenable to trust improvements through public verification, which is the motivation for this work.

The core of the paper proposes cryptographic protocols for public verification of PHM. Section 3 provides technical preliminaries, building on Apple's protocol because it has attracted so much attention, is efficient, and is built on flexible primitives. The concepts that we propose could translate to other PHM constructions. Sections 4, 5, and 6 offer protocols for the following types of verification:

- **Threshold Certification of the Hash Set (Section 4).** A service can prove that the hash set was certified by a specific combination of external groups. These could be child safety groups, such as the National Center for Missing and Exploited Children (NCMEC) in the United States and the Internet Watch Foundation (IWF) in the United Kingdom. The protocol does not require that participating groups share hashes with each other.
- **Proof of Non-Membership in the Hash Set (Section 5).** A service can prove that specific content (e.g., a pro-democracy meme) is not in the hash set, while ensuring that users do not learn anything else about the hash set.
- **Guaranteed Eventual Detection Notification (Section 6).** A service is cryptographically committed to notifying a user if they have content that was revealed as a false positive hash match after a fixed delay (e.g., after a specified period of time or number of images).

Before turning to the ongoing tussles about E2EE and CSAM detection, we emphasize that this project is not an endorsement of PHM systems. We take at face value the claims by law enforcement, child safety groups, and online services that hash matching is a valuable tool for fighting CSAM (e.g., [32]–[34]), and we share the urgency in addressing the proliferation of CSAM on E2EE services.

The protocols that we propose are, however, only trust *improvements*—not trust *solutions*. These forms of verification do not fully address the many cogent critiques of PHM systems [14], [20]–[23], [26], [28]–[30]. We would be deeply uncomfortable about large-scale deployment of automated content detection for E2EE services even with the public verification protocols that we present. At minimum, complementary legal safeguards would be essential to further ensure that PHM systems have narrow scope, operate transparently, and are accountable to the people they affect. Our goal, like prior work on PHM and encryption policy [14], is to advance the policy conversation and the research literature before any deployment—even though we do not yet see a satisfactory system design on the horizon.

2. Benefits, Challenges, and Risks of Hash-Based CSAM Detection Under E2EE

In this section we briefly review the importance of CSAM detection, current hash matching methods for detection in non-E2EE settings, and the risks and challenges of implementing PHM in E2EE settings that we and others have previously articulated (especially [20], [35]–[38]).

2.1. The Importance of CSAM Detection

Online child safety poses a broad range of problems, and CSAM is just one component of the overall landscape [39], [40]. But detecting and taking action against CSAM is particularly urgent, for a number of important motivations.

First, the same people who share or receive CSAM may engage in other forms of child sexual abuse. Some CSAM offenders commit additional "contact" acts of abuse, such as physical molestation or eliciting sexual behavior [41], [42]. Estimates of the share of CSAM offenders who previously committed contact offenses against a child range from 12% to 55% [43], [44]. Locating these CSAM offenders can provide important leads for investigating physical child sexual abuse and rescuing children from dangerous environments.

Second, there is currently a market for CSAM, with an estimated $250 million to $20 billion U.S. dollars spent on CSAM annually [45]. Detecting and preventing the transfer of CSAM dries up the market, lessening financial incentives to create CSAM through physical sexual abuse [45], [46].

Third, survivors of child sexual abuse can be revictimized by the proliferation of recorded imagery of their abuse. A 2017 survey of survivors of child sexual abuse found that 25% reported being identified, propositioned, and blackmailed by people who recognized them from CSAM imagery. 67% of survivors reported trauma related to the permanent nature of the imagery itself [47]. Halting the ongoing distribution of CSAM is essential to prevent perpetuating harms to children who have been exploited.

Finally, preventing the spread of CSAM is a worthy goal unto itself. The content is highly offensive and lacks societally redeeming value. As the U.S. Supreme Court has noted, the public interest in curtailing CSAM is "evident beyond the need for elaboration" [46].

2.2. Hash-Based Matching for CSAM in the Unencrypted Setting

Major social media platforms use a combination of user reports, proactive automated scanning of content, metadata analysis, and other techniques to detect CSAM and other types of abuse. In a recent survey of business practices for addressing problematic user content, involving 13 respondents (from 11 companies), 7 respondents opined that automated monitoring was currently the most effective tool for detecting child sexual abuse imagery [34]. The next highest response was user reporting, identified by just 2 respondents. This result was different for CSAM than for other types of abuse (e.g., hate speech or malware) where the survey respondents considered other techniques to be more effective than automated content scanning.

Public policy discussion and industry practices for automated CSAM detection center on hash matching against known CSAM. In the unencrypted setting, hash matching is the de facto standard for CSAM detection [5], [6], [13]. This is starting to change, and there is momentum toward machine learning classifiers to identify CSAM (e.g., [7]).

Hash-based detection systems typically use a perceptual hash function (PHF) instead of or in addition to a cryptographic hash function (CHF). A PHF, unlike a CHF, yields the same hash or a similar hash when an image is modestly perturbed (e.g., rotation or recompression). See Section 3.1 for details. An online service compares the hash value for sent or stored user content against a database of known harmful media, which could be internal to the service or external (e.g., a third-party API). If there is a match, the service may or may not conduct some additional human review. The service then notifies an appropriate law enforcement agency or child safety group.

In the U.S., online services are required by federal law to promptly report apparent CSAM to the National Center for Missing and Exploited Children (NCMEC), a child safety group that works closely with law enforcement as an information clearinghouse. Specifically, under 18 U.S.C. § 2258A, a company must report to NCMEC any "facts or circumstances . . . from which there is an apparent violation" of federal criminal law prohibiting CSAM "as soon as reasonably possible after obtaining actual knowledge of [those] facts or circumstances."

2.3. Private Hash Matching and Its Limitations

When a user sends or stores data with E2EE, the online service cannot access the user's content. This important security property is an impediment to conventional content-based CSAM detection methods, which has led to immense interest from law enforcement agencies, child safety groups, and online services in alternative detection methods that could operate in E2EE environments.

The primary focus for proactive CSAM detection under E2EE, so far, is private hash matching (PHM).[1] A PHM system has the same detection capabilities as ordinary server-side hash matching, comparing user content to a set of known harmful media, but with an added privacy property: the server only learns about matching content and learns nothing about other content [15], [38], [49].

PHM has the same inherent limitations as ordinary hash matching. These systems only detect known content; they cannot identify new instances of CSAM. Hash matching systems are also simple for an adversary to circumvent by strategically reencrypting or perturbing media.

Despite these limitations, hash matching systems succeed: in 2020, NCMEC received 21.4 million reports of apparent CSAM from online platforms [50]. There are still many people who send and receive known CSAM and do not take steps to conceal the content from online services.

Recent proposals have begun to examine the use of machine learning (ML) methods to detect novel CSAM, such as by implementing client-side classifiers that report results to the server [40], [51]–[53]. The issues that we discuss about PHM for E2EE services also generally apply to ML-based detection systems, in some cases even more so (e.g., heightened false positive rates [40]).

The focus of this work is on transparency improvements for PHM systems that partially address the risks that they pose. While a full treatment of transparency improvements for ML-based systems is beyond the scope of this paper—and an important topic for future work—we note that our transparency protocols related to hash sets (Sections 4 and 5) would not apply to ML-based systems, but our proposal for eventual notification could.

2.4. Risks and Challenges of PHM

We focus on two main difficulties of detecting CSAM via PHM, which motivate the remainder of this work. For a more comprehensive treatment of the risks and challenges of PHM, we refer the reader to [20], [24], [35]–[38].

From the perspective of a user sending messages via an E2EE service that uses PHM to detect CSAM, two essential properties are *trust in the hash set* and *trust in the matching implementation*. As we discuss further below, an untrustworthy hash set or an untrustworthy implementation creates serious risks for security, privacy, and free expression. Improving trust in these areas, through auditability and transparency, should be a priority for PHM systems.

2.4.1. Trust in the hash set. The hash set is the foundation of a PHM system, because it determines the exact set of

1. Another detection mechanism that has generated interest and been deployed, but that is not proactive, is manual user reporting. In a user report, one of the "ends" in the E2EE communication (typically the message recipient) forwards harmful content to the service. A "message franking" protocol can guarantee the authenticity of user reports [48].

content to match. Important challenges include: What goes in the hash set? Who controls it? And, as we focus on in this work, how can the public verify promises made by the curator of a confidential hash set?

In the E2EE setting, a trustworthy hash set is incredibly consequential. The hash set forms, essentially, an exception to the fundamental security and privacy guarantees of E2EE. Put differently, the hash set acts as a sort of key for decrypting a subset of otherwise E2EE communications. These implications make the composition of a hash set extremely important, since the very purpose of a PHM system is to preserve E2EE's benefits with the sole exception of CSAM.

Kamara et al. and Abelson et al., among others, note that hash-based detection of CSAM is *technically* no different from hash-based detection of any other content [20], [24]. The composition of the hash set—no more, no less—is what scopes a PHM system to solely detecting CSAM.

Proponents of PHM systems generally argue for considering these proposals as-is: we should assume that a system will scan only for CSAM and nothing more, and there is a strong moral imperative to help children now (e.g., [18], [40]). Privacy and civil liberties advocates have responded that once a content detection system is available for E2EE communications, there will be immense pressure to increase the scope of the system [14], [20]–[23], [25]–[28], [30], [54]–[56]. Law enforcement agencies worldwide have already stated interest in revealing other information currently hidden by E2EE [17], [57]. Foreign governments have repeatedly pressured U.S. companies to use content moderation systems to censor content within their borders, and firms often bow to this pressure (e.g., [58]–[60]).

The slippery slope debate about PHM systems is, fundamentally, a set of differing predictions about how these new technical capabilities would interact with geopolitics. Borrowing from Volokh's taxonomy of slippery slope mechanisms [61], there are several ways in which a PHM system for CSAM could lead to PHM for other content.

- *Lowering costs:* Designing and implementing a PHM system for detecting CSAM would be a significant fixed cost. After deploying the system, though, the marginal cost of adapting it to detect other categories of content would be modest—just an update to the hash set. In other words, the cost to move an E2EE system from detecting 0 pieces of content to 1 piece of content would be high, but the cost to move the system from detecting n to $(n+1)$ pieces of content would be low.
- *Changing attitudes:* E2EE systems, as presently implemented, presumptively hide all content from all third parties. Adding PHM for CSAM to these systems could reshape perceptions of acceptable security tradeoffs and broadly legitimate the notion of exceptions to E2EE guarantees. Foreign governments might strategically leverage this viewpoint, claiming that if PHM for CSAM is acceptable in the U.S. then PHM for locally objectionable content should also be acceptable. Even if U.S. firms manage to hold out against this pressure, foreign governments would have a strengthened argument for requiring domestic firms to implement PHM.
- *Political momentum:* Implementation of PHM for CSAM could create or reinforce broader political trends toward stricter forms of content moderation [62].

Auditability of the hash set. Robust auditability can improve trust in the hash set, giving users confidence that it consists solely of hashes of known CSAM media. While auditability cannot *prevent* expanding a hash set beyond CSAM, it can provide *notice* of scope creep, enabling responses by users, civil society groups, and governments.[2]

Prior efforts to evaluate and develop consensus about encryption policy have identified auditability as a key goal [20], [35], [36], [40]. Apple itself suggested a third-party auditing mechanism for aspects of its PHM system for CSAM detection, which we touch on in Section 4.

Unfortunately, the goal of hash set auditability runs headlong into *server privacy*, an important property of PHM systems for CSAM detection. Server privacy, which we discuss further in Section 3.2, guarantees that users do not learn the hashes that are in the hash set in order to protect fragile detection methods and sensitive law enforcement investigations. Moreover, users do not (and legally cannot) possess the original CSAM media that led to the hash values in the hash set. Users, as a result, cannot simply inspect a CSAM hash set and confirm its composition even if the hashes were public.

While users cannot directly examine hashes in the hash set, there are next-best forms of auditability that are feasible. First, third parties can, individually or in combination, provide attestation about the hashes that are in the hash set. We propose a protocol for this threshold external validation in Section 4. Second, an online service can provide proof that particular hashes are *not* in the hash set. We present a protocol for proof of hash set non-membership in Section 5.

The benefits of public verification. There is broad agreement that, where feasible, it is preferable to enable *users as verifiers* as a complement to third-party auditing. Abelson et al. note the desirability of deploying systems that are "predict[able] and audit[able] *by users*" (emphasis added) [20]. Canetti and Kaptchuk suggest having "user's phones" verify the validity of digests beyond third-party audits [30]. Levy and Robinson propose addressing risks of scope creep with "cryptographic assurances [on the hash database] which can be attested to *publicly* and audited privately" [40] (emphasis added).

In the strongest form, public verification enables users to directly prove a property of the hash set. A manipulation of the hash set would (assuming correct implementation) necessarily alert users. In a weaker but still very valuable form, public verification enables users to obtain proof that third parties verified specific hash set properties. This form of verification acts as a delegation of trust in the hash set, shifting the locus of trust from the online service that operates the PHM system and curates the hash set to the third parties that originate the hashes in the hash set. An

2. A client implementation of PHM could make protocol participation contingent on passing public verification. Disabling that verification would, however, be technically trivial—effectively rendering it a form of notice.

adversary would have to compromise or corrupt sufficient parties to manipulate the hash set without alerting users.

Public verification has other important benefits. It allows many parties to conduct verification, including highly motivated parties like privacy and civil liberties groups, increasing the likelihood of rapidly identifying any misconduct. Verification can also be an ongoing process, instead of waiting potentially long periods between occasional audits. This property avoids gaps in verification coverage and could enable faster updates to the hash set.

In Sections 4 and 5, we describe two protocols for public verifiability of the hash set. We emphasize again that these protocols are improvements for trust in the hash set, not complete solutions. The protocols provide early warning of hash set manipulation—they do not prevent hash set manipulation. Furthermore, as we discuss in the following subsection, cryptographic improvements do not address the challenge of ensuring that an online service has faithfully implemented a PHM system for CSAM detection.

2.4.2. Trust in the implementation. Suppose that an online service has curated a hash set that consists solely of hashes derived from CSAM media. A user would still have cause for concern: the PHM system's implementation of matching against the hash set could flag content that is not CSAM or otherwise expose the user's content to third parties. This problem is particularly acute for PHM deployments in E2EE settings, because it implies plaintext access to user content that should have remained encrypted and confidential.

False positives are one important way in which a PHM system's implementation could be untrustworthy. These are instances where the PHM system identifies a match even though a user's content is perceptually dissimilar to the original CSAM media for the hash set. The cryptographic components of PHM systems induce, depending on the construction, either a negligible false positive rate [15] or no false positives [14]. The underlying perceptual hash function can, however, introduce false positives that are difficult to empirically characterize. Prior work has suggested PHF false positive rates ranging from 1 in 1,000 [63] to (in Apple's proposal) 3 in 100 million [64]. These estimates are sensitive to assumptions about hash set composition and user content.

Adversarial false positives further complicate the situation. A malicious party could induce a false positive by generating an image that is visually innocuous but that has the same perceptual hash value as a known CSAM image. Prior work on encryption policy and PHM systems elaborates on this risk in further detail [20], [24]. There is a vast literature on using machine learning to generate adversarial examples (see [65]), including for the NeuralHash PHF that Apple proposed to use with its PHM system [66]–[68]. Levy and Robinson note, though, that there appear to be "no real-world [adversarial ML] attacks against current perceptual hash systems in over a decade of widespread use" [40].

The software and hardware security of PHM systems is another cause for concern. Adding PHM to an online service inherently expands the threat surface and could introduce new vulnerabilities. Abelson et al. [20] discuss specific ways in which PHM could create security risks, depending on where the scanning code is located, who writes the code, and where reports are sent. The security risks of code complexity are compounded by the difficulty of verifying the behavior of client and server code.

Auditability of the implementation. Public verification of individual false positives is a promising direction, providing notice to affected users and enabling overall accountability for PHM systems. In other areas of encryption poliicy, stakeholders have previously agreed that informing affected users is appropriate for due process and oversight [30], [35], [36]. In Section 6, we present a protocol that ensures a user receives eventual notice after their content is flagged as a positive match but an online service takes no action because the content is a false positive.

While not the focus of this work, we note that aggregate forms of public verification for false positives would also be feasible. A version of the protocol that we propose, for example, could enable a periodic count of false positives across all users without revealing individual false positives. For ordinary hash-matching systems, it may be possible to statically compute a false positive rate using functional encryption or verifiable computation.

As for the security risks from implementing PHM, there are known—though nontrivial to implement—methods for program verification that could improve trust [69]–[72]. Applying these methods would help prevent bugs, insider attacks, or intentional alterations to a PHM system.

The foundational motivation of this paper is that maximizing cryptographic improvements to trust—in the hash set and in the implementation—will be essential to exploring any possible paths forward for PHM systems. To that end, in the balance of the paper, we present three protocols for improving trust: threshold zero-knowledge certification of the hash set (Section 4), proof of non-membership in the hash set (Section 5), and guaranteed eventual notification of detection (Section 6).

3. Technical Preliminaries

In this section we first provide technical definitions that we use in later sections (Section 3.1), then we offer a brief overview of the Apple PHM proposal that we extend with our protocols (Section 3.3), and finally we touch on how we respected ethics in carrying out this work (Section 3.4).

3.1. Technical Definitions

We denote a field of prime order p as $\mathbb{F}_p$. We notate an elliptic curve over field $\mathbb{F}$ as $E(\mathbb{F})$. We use $[n]$ as shorthand for $\{1, \ldots, n\}$. A *negligible* function $\mathsf{negl}(\cdot)$ is one such that for every positive polynomial function $\mathsf{poly}(\cdot)$, there exists some $n > 0$ such that for all $x > n$, we have $\mathsf{negl}(x) < \frac{1}{\mathsf{poly}(x)}$. Let $\epsilon_{\mathsf{coll},n,q}$ be the chance of a collision occurring among n randomly sampled elements from a field of size q; if n is polynomial and q is super-polynomial in a security parameter then $\epsilon_{\mathsf{coll},n,q}$ is negligible in the parameter. We proceed to define several useful cryptographic tools:

Definition 1 (Threshold Signature Scheme (adapted from [73], [74]))**.** A threshold signature scheme is a set of algorithms $\mathscr{T} = (\mathsf{TKeygen}, \mathsf{TKeyCombine}, \mathsf{TSignShare}, \mathsf{TVerShare}, \mathsf{TCombine}, \mathsf{TVerify})$ among N parties known in advance, with threshold $1 \leq \tau \leq N$, so that, informally:

1) $\mathsf{TKeygen}() \to ((\mathsf{sk}_1, \mathsf{vk}_1), \ldots, (\mathsf{sk}_N, \mathsf{vk}_N))$ is a (possibly interactive) method that grants each party possesses a secret key share sk_i and public verification key share vk_i.
2) $\mathsf{TKeyCombine}(\mathsf{vk}_1, \ldots, \mathsf{vk}_N) \to \mathsf{vk}$ is a function that produces the aggregate public verification key vk from the public verification key shares vk_i.
3) $\mathsf{TSignShare}(\mathsf{sk}_i, m) \to \sigma_i$ yields a signature share on m for party i.
4) $\mathsf{TVerShare}(\mathsf{vk}_i, \sigma_i', m)$ returns 1 if σ_i' is a valid signature share, 0 otherwise
5) $\mathsf{TCombine}(A, m) \to \sigma_*$ yields an aggregate signature on m if A contains valid signature shares from at least τ different parties.
6) $\mathsf{TVerify}(\mathsf{vk}, \sigma_*', m)$ returns 1 if aggregate signature σ_*' is a valid re-combination of at least τ valid signature shares on m, 0 otherwise.

The scheme has individual and aggregate *correctness* and *existential unforgeability under chosen message attack*.

Informally, unforgeability means that it is computationally difficult to create a signature which passes $\mathsf{TVerify}$ without access to at least τ secret key or signature shares. The standard formalization is omitted for space; see [73]–[75]. Note also that the aggregate signature is the same no matter which τ or more parties' shares contribute to the aggregate signature. See [75], [76] for additional discussion.

Definition 2 (Zero-knowledge Proof of Knowledge)**.** The protocol $(\mathcal{P}(x, w), \mathcal{V}(x))$ is a *zero-knowledge proof of knowledge* for the NP relation $R(x, w)$ if the following properties are met:

- *Completeness*: If $\mathcal{P}$ and $\mathcal{V}$ are honest and $R(x, w) = 1$, $\mathcal{V}$ always outputs Accept when interacting with $\mathcal{P}$.
- *Soundness*: For any malicious prover $\mathcal{P}^\star$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that if $R(x, w) = 0$, $\mathcal{V}(x)$ outputs Reject with probability at least $1 - \mathsf{negl}(|x|)$ when interacting with $\mathcal{P}^\star$.
- *Knowledge soundness*: If a computationally-bounded malicious prover $\mathcal{P}^\star$ can interact with $\mathcal{V}(x)$ and cause it to output Accept with non-negligible probability, then there exists an extractor $\mathcal{E}^{\mathcal{P}^\star}(x)$ that can output a witness w such that $R(x, w) = 1$.
- *Zero-knowledge*: There exists a probabilistic polynomial-time machine Sim such that for every polynomial-time algorithm $\mathcal{V}^\star$, $\mathcal{V}^\star(x, z)$'s output when interacting with $\mathcal{P}(x, w)$ is indistinguishable from $\mathsf{Sim}^{\mathcal{V}^\star(x, z, \cdot)}$ for $R(x, w) = 1$ and $z \in \{0, 1\}^*$.

Definition 3 (Homomorphic Commitment)**.** A homomorphic commitment scheme $(\mathsf{HCom}, \mathsf{HDecom}, \mathsf{HAddCom})$ over a finite field $\mathbb{F}_q$ allows a sender to produce a commitment $C = \mathsf{HCom}(x, r)$ in order to commit to $x \in \mathbb{F}_q$ with randomness $r \in \mathbb{F}_q$. Later, the sender can reveal $\mathsf{HDecom}(C, x, r)$ which "decommits" C and either returns Valid ($\top$) or Invalid ($\bot$). Informally,

- (Computational) *Binding*: A (computationally bounded) adversary cannot produce C, x, x', r, r' such that $\mathsf{HDecom}(C, x, r)$ and $\mathsf{HDecom}(C, x', r')$ are both valid, where $x \neq x'$.
- (Computational) *Hiding*: A (computationally bounded) adversary given access to C cannot produce x.
- *Homomorphism*: For all pairs of messages x_1, x_2 in the message space, and all pairs of randomnesses r_1, r_2 in the randomness space, where $C_1 = \mathsf{HCom}(x_1, r_1)$ and $C_2 = \mathsf{HCom}(x_2, r_2)$, we have

$$\mathsf{HAddCom}(C_1, C_2) = \mathsf{HCom}(x_1 + x_2, r_1 + r_2).$$

We also informally define the following tools, which we make use of in our protocol constructions.

IND$-CPA-secure Encryption with Random Key Robustness. In Section 4 we use an encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ with the same properties as in [15]. In short, the scheme must be both indistinguishable against random chosen plaintext attack, and "random key robust," meaning that if $c \leftarrow \mathsf{Enc}(k, m)$, then if $k' \leftarrow \mathcal{K}'$ is a random key then the decryption $\mathsf{Dec}(k', c)$ fails with high probability. See [15] for a more detailed definition.

Private Set Intersection. Private Set Intersection (PSI) is a Multi-Party Computation protocol between two parties A and B. Party A possesses set X, and party B possesses set Y. The PSI functionality informally allows A and B to learn the intersection $X \cap Y$ without learning anything else about each other's sets. Much prior work in PSI exists, see [15] for an explanation of Apple's specific setup.

Cuckoo Tables. Cuckoo tables have $k \geq 2$ hash functions $h_1, \ldots, h_k$ (in this work, $k = 2$) mapping the input universe to the size of the data structure. Cuckoo tables provide the guarantee that if element y is in the table, it is located at one of $h_1(y), \ldots, h_k(y)$. For further discussion we refer the reader to [77]–[79]. There is a robust line of work using Cuckoo tables in PSI, including [80]–[84].

Perceptual Hash Functions. A Perceptual Hash Function (PHF) is informally a type of hash function that produces similar or identical hashes for perceptually similar inputs—inputs that a person would consider to be the same image. Perceptual hashes on images are designed to be robust to common image perturbations such as scaling, color or brightness changes, transformations, rotations, watermarks, or noise [85]–[90]. The hashes of similar inputs are similar with respect to a distance metric, typically Hamming distance. Note that PHFs are quite different from cryptographic hash functions, which aim to achieve collision resistance and unpredictable output regardless of input similarity. PHFs do not (by design) exhibit an avalanche effect, and they are vulnerable to collision, preimage, and second-preimage attacks [91].

3.2. Goals and Methods for Private PHM

Current proposals for CSAM detection in E2EE online services [14], [15] detect CSAM in the following way. The

service holds a set X of perceptual hashes of known CSAM. The user has some message y. The goal of PHM is to allow a party, usually the server, to detect if $y \in X$ (for exact matching[3]) while maintaining the following properties.

- *Client privacy*: The server learns no information about a user's hash y unless it is in the server's hash set X.
- *Server privacy*: The user learns no information about the hash set X other than its size $|X|$ from the PHM protocol. The user may learn the results of the server's actions after the PHM protocol, such as if the user's image or account is blocked based on PHM output.

Constructions of PHM for CSAM attempt to achieve both properties. In contrast, other moderation systems deployed at scale may not have client privacy (possibly because the server already has access to the user's plaintext) or server privacy (because the server's list is not sensitive).

3.3. The Apple PSI System

Here we describe the Apple PSI system—Apple's PHM proposal—at a high level. We refer readers to the technical summary [92] and Bhowmick et al. [15] for further detail.

In Apple's *fuzzy threshold PSI with associated data*, the client has a list of triples each containing a hash, a random ID (a randomly chosen public identifier for the triple), and some associated data. Apple has a set X of hashes of CSAM. After running the protocol, if the client has fewer than a threshold t_{sh} of elements whose hashes match Apple's set, Apple's goal is to learn only the IDs of the triples, and nothing else. If the client has at least t_{sh} matching elements, Apple wishes to additionally learn the associated data of all matching elements.

Public parameters of the Apple PSI System include an elliptic curve group $E(\mathbb{F}_p)$ with public generator G, a hash function $H : \{0,1\}^* \to E(\mathbb{F}_p)$, a random-key-robust symmetric encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ with keyspace $\mathcal{K}'$, as well as a KDF $H' : E(\mathbb{F}_p) \to \mathcal{K}'$.

Using hash functions h_1 and h_2, the server arranges hash set X into a Cuckoo table T of length n, where $n > |X|$. Empty slots are filled with random *dummy elements* (which will require special treatment in Section 4). The server chooses blinding key $\alpha \in \mathbb{F}_p$ and publicly releases $\mathsf{pdata} = (L, \{P_j\}_{j\in[n]})$ where

$$L = \alpha \cdot G,$$
$$P_j = \alpha \cdot H(T[j]).$$

The client's set Y contains images to match against X. Each element has *associated data* ad, such as a small grayscale version of the image. Semi-honest clients[4] facilitate the matching process by sending a *voucher* for each set element.

This voucher contains $\mathsf{Enc}(\mathsf{adkey}, \mathsf{ad})$, and the adkey will be revealed to Apple if and only if the client's set Y has at least t_{sh} matches with Apple's hash set X. To accomplish this, with each image a t_{sh}-out-of-t_{sh} Shamir share of adkey is also sent to the server, encrypted with an ephemeral key $rkey$. The clients additionally send four additional objects defined below, with 1 and 2 shuffled: Q_1, Q_2, $\mathsf{Enc}(H'(S_1), rkey)$, and $\mathsf{Enc}(H'(S_2), rkey)$, where for $j \in \{1,2\}$, and for random $\beta_j, \gamma_j \in \mathbb{F}_p$:

$$Q_j = \beta_j \cdot H(y) + \gamma_j G$$
$$S_j = \beta_j \cdot P_{h_j(y)} + \gamma_j L$$

Observe that a computationally bounded server is able to decrypt $rkey$ (and therefore learn an additional Shamir share of adkey) if and only if $P_{h_j(y)} = \alpha H(y)$ for one of $j = 1$ or $j = 2$, meaning that y must occupy one of its two possible locations $h_1(y), h_2(y) \in \{1, \ldots, n\}$ in the Cuckoo table. If the server cannot decrypt either encryption of $rkey$, it considers the image a non-match. Each match sent by the client allows the server to learn one Shamir share of adkey. If the client sends t_{sh} many vouchers that do match, the server reconstructs adkey and can then decrypt the associated data in all matching vouchers sent thus far. We call this event a "positive" for the remainder of this work; it may be a true or false positive.

Finally, to hide the exact number of matching vouchers, the *fuzzy* part of the protocol is that the client occasionally sends "synthetic matches" in which the server can obtain $rkey$ but only learns a "dummy" share of adkey. This component of Apple's protocol is orthogonal to our work.

3.4. Ethical Considerations

CSAM detection is an exceptionally sensitive topic. We took several steps that we believed were appropriate to responsibly contribute in this area. First, we briefed firms that were considering implementing PHM systems (including Apple) on this line of research and how to practically implement it. We sought to ensure that any forthcoming PHM deployments would include trust improvements for the benefit of users and society. Second, we closely engaged with other encryption policy stakeholders to inform this work, including law enforcement agencies, child safety groups, and privacy and civil liberties organizations. We also filed comments on the evaluation criteria for the U.K. government's initiative to fund CSAM detection systems [19]. Third, we relied exclusively on random or non-sensitive media in carrying out this research. At no time did we have access to or seek access to CSAM content or hashes.

4. Threshold Zero-Knowledge Certification of the Hash Set

As discussed in Section 2.4, a key concern about private hash matching is *trust in the hash set*. Even if the firm curating the hash set is trustworthy, it could face immense pressure to expand the set to include material outside the original purpose of the deployment. In Apple's case, concerns about this pressure included both domestic content

3. We focus on exact matching PHM systems in this paper, but our protocols could be adapted to approximate matching systems which identify a match if there is a hash $x \in X$ where x is sufficiently similar to y.

4. Apple's protocol maintains server privacy against malicious clients, but can only guarantee correctness against semi-honest clients [15].

moderation (e.g., other types of problematic or unlawful content) and foreign governments that had already attempted to force Apple's hand in other policy issues [58], [60], [93].

Apple's planned approach to handling this pressure was to take the following two steps. First, Apple stated that its hash set would be built exclusively from elements that were provided by at least two independent child safety groups (e.g., NCMEC in the U.S. and IWF in the U.K.). Second, Apple sketched a high-level procedure by which a third-party auditor might verify both the hash set itself and the public information pdata, which is used to detect matches with client content. The pdata contains a commitment $L = \alpha \cdot G$ to Apple's secret blinding key α and a blinded Cuckoo table containing elements $P_j = \alpha \cdot H(T[j])$ where T is a Cuckoo table of CSAM perceptual hashes and H is a hash onto an elliptic curve.

Apple provides only sparse details in their technical specification [15] and threat model document [64] of what this auditing might look like and how it would verify the threshold intersection property. They state first that child safety groups could provide a "non-sensitive attestation of the full database they sent to Apple" and Apple could provide "technical proof to the auditor that the [threshold] intersection and blinding were performed correctly" [64]. Here Apple is presumably referring to a cryptographic proof that would demonstrate that the hash set X was built only from elements appearing in at least two child safety groups' lists, based on those groups' attestations. On page 13, Bhowmick et al. [15] also present a high-level non-cryptographic approach to verifying that the pdata was correctly formed from X, but their description suggests that X would have to be known to the auditor.

For the reasons discussed in Section 2.4.1, it would be valuable to complement auditing with public user verification. In this section we describe a cryptographic method by which users can directly and frequently verify that the threshold intersection and blinding was done correctly. At the same time, the protocol will not require the child safety organizations or users to learn any additional information.

An ideal tool to use in this scenario is a *threshold signature scheme*: a threshold signature on a message can be verified using a public key as long as at least τ-out-of-N signing groups contribute signature shares. For more details see Definition 1 and [73]–[76]. A naive implementation in which child safety groups directly build threshold signatures on each pdata element would, however, present the following two challenges.

The first challenge is mainly technical: The users have access to pdata elements $P_j = \alpha \cdot H(T[j])$, but do not know the blinding key α or the elements $T[j]$ themselves. The only "message" available over which a user could verify a signature is the pdata element P_j. On the other hand, although the child safety organizations know the elements themselves and could verify elements $T[j]$ as being on their lists of known CSAM, the child safety organizations do not know Apple's blinding key α. Due to the hardness of Decisional Diffie Hellman in the elliptic curve group, without knowledge of α the child safety groups cannot distinguish pdata elements like $P_j = \alpha \cdot H(T[j])$ from random and therefore cannot directly certify that P_j corresponds to known CSAM element $T[j]$. The main purpose of our new Protocol Π_{cert}, then, is to create a threshold signature on pdata element P_j if and only if $T[j]$ is held by at least τ child safety organizations.

The second issue has to do with the random dummy elements. In Apple's scheme, after building Cuckoo table T from set X, there will be empty slots remaining in T; Apple fills these slots with random "dummy" elements. Even without cryptographic verification, it is not clear how a third-party auditor could verify that these elements are actually random and do not represent other random-looking hashes of meaningful content. Apple's only description of how these would be audited is to say that they "treat this [non-random choice of dummies] the same way as one would treat a malicious server who is attempting to modify the provided set X" [15]. At this high level it is not clear how an auditor would be able to verify after the fact that a given elliptic curve point was randomly chosen compared to it being the hash of some non-random element. In our scheme, we will have Apple and the child safety groups collectively generate a random seed that will be used as the key to a PRF, and that PRF will be used to generate the dummy elements pseudorandomly rather than randomly. This also allows the child safety organizations to create signatures on the pseudorandom dummy elements with confidence that the dummy elements are not meaningful.

We proceed to describe Protocol Π_{cert}. The exact properties of our protocol are stated informally in more detail in Section 4.1, and are stated and proven formally in Section 4.2. Ultimately, this protocol cannot eliminate dependence on some final root of trust, nor does it guarantee that all of the content contained in the pdata is truly CSAM. It does, however, raise the bar: it reduces the amount of trust placed in Apple alone, delegates and distributes that trust among child safety groups whose public interest incentives pull them in different directions, requires any external pressure to expand the list to corrupt τ of these groups, and allows users to validate the hash set under those assumptions much more frequently than third-party audits would.

4.1. Protocol Goals and Overview

Protocol Π_{cert} (shown in Figure 1a-1b) is a multi-party protocol among a server $\mathcal{S}$, N child safety groups $\mathcal{G}_1, \ldots, \mathcal{G}_N$ (which we often call simply "groups"), and a user $\mathcal{U}$. The goal of the protocol is for the server $\mathcal{S}$ and groups to work together to prove to $\mathcal{U}$ that each element of the pdata was built either from a hash held by at least τ child safety groups, or is pseudorandom, and we accomplish this without sacrificing server privacy.

As we will prove in Section 4.2, for particular sets of party corruptions, Π_{cert} implements functionality $\mathcal{F}_{\mathsf{cert}}$, which has the following informal properties:

Unforgeability. Informally, even if the server and $d < \tau$ malicious groups collude, these parties cannot cause improper verification of a hash not contributed by at least

Protocol Π_{cert} (Part 1)

Public parameters: Number of groups N, threshold τ, Cuckoo table parameter ϵ', generator G of elliptic curve group $E(\mathbb{F}_p)$, prime q, hash set element universe $\mathcal{E}$, seed size ℓ, pseudorandom function $\mathsf{PRF} : \{0,1\}^\ell \times \mathbb{N} \to \mathcal{E}$, hash function $H : \mathcal{E} \to E(\mathbb{F}_p)$ KDF $H' : E(\mathbb{F}_p) \to \mathcal{K}$ where $\mathcal{K}$ is the keyspace of $(\mathsf{Enc}, \mathsf{Dec})$.
Sub-functionalities: $\mathcal{F}_{\mathsf{coin}}$ (a standard coin-flipping functionality, see Figure 5
Inputs: Each group $\mathcal{G}_i$ inputs a set X_i. $\mathcal{S}$ inputs n, the eventual size of the Cuckoo table. User $\mathcal{U}$ has no input.
Establishing the Cuckoo table:

1) (Setup.) Server $\mathcal{S}$ randomly chooses blinding key $\alpha \leftarrow \mathbb{F}_q$.
   a) (Threshold signature keygen.) Groups $\mathcal{G}_1, \ldots, \mathcal{G}_N$ run the threshold signature scheme's $\mathsf{TKeygen}$. Each group $\mathcal{G}_i$ broadcasts its verification key share vk_i. $\mathcal{U}$ recreates $\mathsf{vk} \leftarrow \mathsf{TKeyCombine}(\mathsf{vk}_1, \ldots, \mathsf{vk}_N)$.
   b) (Generating seed.) $\mathcal{S}$ and the groups $\mathcal{G}_1, \ldots, \mathcal{G}_N$ collectively run $\mathcal{F}_{\mathsf{coin}}$ to generate a shared ℓ-bit seed seed.
2) (Building X.) Each group $\mathcal{G}_i$ sends its set X_i to $\mathcal{S}$. $\mathcal{S}$ builds the set X of elements appearing in at least τ of these sets: $X = \{e : \exists I \subseteq [N] \text{ s.t. } |I| \geq \tau \wedge e \in X_i \forall i \in I\}$.
3) (Building the Cuckoo table.) $\mathcal{S}$ does the following:
   a) Compute $L = \alpha \cdot G$. Let $n = \epsilon'|X|$.
   b) Pick and tweak random hash functions $h_1, h_2 : \mathcal{E} \to [n]$ where no $e \in \mathcal{E}$ has $h_1(e) \neq h_2(e)$ (see [15]).
   c) Build the initial Cuckoo table T' of length n using a deterministic procedure using h_1, h_2, and X.
   d) The final Cuckoo table T is set as follows: For every entry j of T', if $T'[j]$ is nonempty, then $T[j] = T'[j]$. If $T'[j]$ is empty, then $T[j] = \mathsf{PRF}(\mathsf{seed}, j)$.
   e) For each $j \in [n]$, set $P_j = \alpha \cdot H(T[j])$.
   f) $\mathcal{S}$ publishes to the user $\mathcal{U}$ and groups $\mathcal{G}_1, \ldots, \mathcal{G}_N$ the following: $h_1, h_2, L, (P_1, \ldots, P_n)$.

Figure 1a. First part of Protocol Π_{cert} for ideal functionality $\mathcal{F}_{\mathsf{cert}}$ (Figure 2) in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model. See part 2 of Π_{cert} in Figure 1b, and see Lemmas 5 and 4 for proofs of security properties.

Protocol Π_{cert} (Part 2)

Group certification of table elements:

4) (Sending encrypted signature shares.) Each group $\mathcal{G}_i$ computes $J_i = \{(h_1(e), e) : e \in X_i\} \cup \{(h_2(e), e) : e \in X_i\}$, the set of tuples of that group's elements and their possible locations in T. For each $j \in [n]$, group $\mathcal{G}_i$ acts as follows:
   a) Generate $\sigma_{i,j} \leftarrow \mathsf{TSignShare}(\mathsf{sk}_i, P_j)$, a share of a signature on message P_j.
   b) Let $d_j = \mathsf{PRF}(\mathsf{seed}, j)$ be the dummy value for j. Let $V_{i,j}$ be the set $\{e : (j, e) \in J_i\} \cup \{d_j\}$; this is the set of elements $\mathcal{G}_i$ will approve at index j (the dummy d_j, and any element $e \in X_i$ such that $h_1(e) = j$ or $h_2(e) = j$).
   c) Initialize the empty list $E_{i,j}$, which will contain several encryptions of $\sigma_{i,j}$ under different keys.
      i) Compute $Q_{i,j,k} = \beta_{i,j,k} \cdot H(e) + \gamma_{i,j,k} \cdot G$, where $\beta_{i,j,k}, \gamma_{i,j,k} \xleftarrow{\$} \mathbb{F}_q$.
      ii) Let $c_{i,j,k} \leftarrow \mathsf{Enc}(H'(S_{i,j,k}), (j, \sigma_{i,j}))$ be an encryption of message $(j, \sigma_{i,j})$ under key $H'(S_{i,j,k})$, where $S_{i,j,k} = \beta_{i,j,k} \cdot P_j + \gamma_{i,j,k} \cdot L$.
      iii) Append $(j, Q_{i,j,k}, c_{i,j,k})$ to $E_{i,j}$; note that $\mathcal{S}$ can decrypt $c_{i,j,k}$ if $P_j = \alpha \cdot H(e)$.
   d) Send $E_{i,j}$ to $\mathcal{S}$.
5) (Signature aggregation.) For each index $j \in [n]$ $\mathcal{S}$ does:
   a) Initialize A_j as an empty set. For each group $i \in [N]$:
      i) For each $(j, Q_{i,j,k}, c_{i,j,k})$ in the received $E_{i,j}$:
         A) Compute $S'_{i,j,k} = \alpha \cdot Q_{i,j,k}$. If $Q_{i,j,k}$ was built from e and $P_j = \alpha \cdot H(e)$, then $S'_{i,j,k} = S_{i,j,k}$.
         B) Call $\mathsf{Dec}(H'(S'_{i,j,k}), c_{i,j,k})$, parse the resulting plaintext as $(j, \sigma'_{i,j})$ if decryption did not fail.
         C) Attempt verification of $\sigma'_{i,j}$ by running $\mathsf{TVerShare}(\mathsf{vk}_i, \sigma'_{i,j}, P_j)$.
      ii) If $\mathcal{S}$ is honest and $\mathcal{G}_i$ was honest, then there will be either zero or one element of $E_{i,j}$ such that $\sigma'_{i,j} = \sigma_{i,j}$ verifies (one if $T[j] \in X_i$ or is a dummy, zero if $T[j]$ is not a dummy and not in X_i). If there was such an element, add it to A_j.
   b) Reconstruct $\sigma_{*,j} \leftarrow \mathsf{TCombine}(A_j, P_j)$, the aggregate signature on P_j, using the shares in A_j. Publish $\sigma_{*,j}$.

User verification:

6) For all $j \in [n]$, $\mathcal{U}$ runs $\mathsf{TVerify}(\mathsf{vk}, \sigma_{*,j}, P_j)$. $\mathcal{U}$ outputs 1 if all n signatures verify, 0 otherwise.

Figure 1b. Continuation of Π_{cert} beginning in Figure 1a

$(\tau - d)$ honest groups (i.e. τ total certifications). We note that our protocol does not (indeed, cannot) prevent malicious groups from sharing their hash sets with each other to inflate the number of groups that hold a particular hash, nor colluding with a malicious server to directly send signature shares of $P_j = \alpha \cdot H(e_j)$ even though they do not have the underlying element e_j. However, our protocol does ensure that if d groups are dishonest, at least $(\tau - d)$ honest groups must certify a particular element in order to have it pass verification. This property is shown formally in Lemma 4.

Server Privacy. We also wish to ensure that our scheme has the same privacy properties as Apple's original protocol [15]. Our protocol maintains server privacy, which is formally captured in Lemma 5. Informally, under the same computational assumptions of the original protocol [15] plus a pseudorandom function PRF, a coalition of up to $(N - \tau)$ malicious groups and semi-honest user $\mathcal{U}$ learn no new information as a result of running Π_{cert} aside from the outputs of $\mathcal{F}_{\mathsf{cert}}$ (the hash functions h_1 and h_2, the size of the Cuckoo table n, and the fact that all signatures were certified by τ groups) if all signatures verify. In particular no party except the server $\mathcal{S}$ learns any information about the elements themselves, including which groups certified which elements and which elements are dummies.

This protocol's client privacy is negligibly close to the client privacy guarantee of Apple's protocol; the only change is that the dummies are now pseudorandom.

The server learns the same information it would have learned in Apple's original protocol: the individual hash set X_i of each group $\mathcal{G}_i$. One could design an alternate version of this protocol in which the server does not learn X_i, and learns only the threshold intersection. We choose to allow the server to learn each group's hash set as in the original protocol, to make it easier for $\mathcal{S}$ to catch malicious groups trying to include non-CSAM hashes.

Functionality $\mathcal{F}_{\mathsf{cert}}$

Establishing the Cuckoo table:

1) Provide sub-functionality $\mathcal{F}_{\mathsf{coin}}$ as in Figure 5. Respond to messages from parties until the honest parties have built an ℓ-bit random seed as in the protocol step 1b.
2) From each group $\mathcal{G}_i$, receive set X_i. Send all X_i to $\mathcal{S}$.
3) Receive h_1, h_2, and the Cuckoo table T of length n from $\mathcal{S}$. If Apple was honest then T was built from $X = \{e : \exists I \subseteq [N] \text{ s.t. } |I| \geq \tau \wedge e \in X_i \forall i \in I\}$ (the set of elements appearing in at least τ groups' sets) and seed as in steps 3a-3d of Π_{cert}. Send h_1, h_2, and n to all groups and $\mathcal{U}$. Abort if there are any repeat elements of T.

Group certification of table elements:

4) Process the table in the following way:
   a) For each $j \in [n]$: Receive $(\mathsf{Vouch}, j, V_{i,j})$ from each group $\mathcal{G}_i$. For honest parties, $V_{i,j}$ is the set including $\mathsf{PRF}(\mathsf{seed}, j)$ and any element $e \in X_i$ such that $h_1(e) = j$ or $h_2(e) = j$.
   b) For each $j \in [n]$: Receive $(\mathsf{PublishSig}, j, s_j)$ from $\mathcal{S}$ where $s_j \in \{0, 1\}$. In the honest execution, all $s_j = 1$; if $s_j = 0$ this represents a malicious $\mathcal{S}$ refusing to publish a valid hash certification even if it could do so.

User verification:

5) For each element $j \in [n]$: Send $\mathcal{U}$ $(\mathsf{Verified}, j, 1)$ if both of the following properties are met, else send $(\mathsf{Verified}, j, 0)$.
   a) $T[j]$ appears in at least τ groups' $V_{i,j}$ sets sent as $(\mathsf{Vouch}, j, V_{i,j})$ in step 4a.
   b) and $s_j = 1$ from step 4b
6) An honest $\mathcal{U}$ outputs 1 if it received $(\mathsf{Verified}, j, 1)$ for all $j \in [n]$, 0 if it received any $(\mathsf{Verified}, j, 0)$.

Figure 2. Ideal functionality $\mathcal{F}_{\mathsf{cert}}$ for verifying that at least τ external groups approve of the released pdata.

4.1.1. Certification Protocol Overview. The full protocol Π_{cert} is given in Figure 1a and 1b. A corresponding ideal functionality $\mathcal{F}_{\mathsf{cert}}$ is given in Figure 2; in Section 4.2 we show that the two are indistinguishable under specific assumptions and sets of corrupt parties. Here we provide a brief overview of how the protocol works.

Part 1. Establishing the Cuckoo table: The groups and server begin by generating relevant keys, and building a shared random seed which will be used to generate dummies. The groups send their sets X_i to the server, which builds the threshold set X as the set of all elements appearing in at least τ of the X_i sets. The server builds a Cuckoo table from X, filling the empty spaces in with dummies pseudorandomly derived from seed.

Part 2. Group certification of table elements: Each child safety group $\mathcal{G}_i$ computes a set of elements it would certify for each index j in the pdata, including the pseudorandom dummy element and any element $e \in X_i$ for which $h_1(e) = j$ or $h_2(e) = j$. It generates a signature share $\sigma_{i,j}$ on pdata element P_j, and encrypts it once per each element e of that set. Each encryption uses a key generated from $H(e)$, similar to how the client vouchers are encrypted in [15]. The server will be able to decrypt that signature share if $P_j = \alpha \cdot H(e)$, that is, if the pdata element P_j was the server's blinding key α times the hash of that element onto the elliptic curve. If $P_j \neq \alpha \cdot H(e)$, the server will be unable to decrypt the signature share with high probability. The group sends the sets of encrypted signature shares to the server, who decrypts any shares encrypted using e for which $P_j = \alpha \cdot H(e)$. The server aggregates the shares and publishes the aggregate signature on P_j.

Part 3. User verification: The user verifies each of the threshold signatures and outputs 1 if all signatures pass.

4.2. Proofs of Security

Our proofs are simulation-based proofs in the Simple Universal Composability (SUC) model of Canetti, Cohen, and Lindell [94], like the Apple PSI protocol [15].[5]

In this model, an environment $\mathcal{Z}$ will monitor either a "real" protocol Π, in which it is interacting with real adversary $\mathcal{A}$, or an ideal functionality $\mathcal{F}$, in which it interacts with a simulator Sim. The environment's goal is to distinguish between these settings, outputting a bit 0 or 1. We notate the following:

- $\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$ is the output of $\mathcal{Z}$ in the "real" setting, where several parties (some of whom are controlled by $\mathcal{A}$) conduct protocol Π. $\mathcal{Z}$ sets the inputs and reads the outputs for all parties in the protocol, and can send messages to the real adversary $\mathcal{A}$.
- $\text{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}$ is the output of $\mathcal{Z}$ in the "ideal" setting, where the ideal honest parties are interacting with $\mathcal{F}$, and Sim is "simulating" the dishonest parties (by running a copy of $\mathcal{A}$). As in the real setting, $\mathcal{Z}$ sets inputs and reads outputs of all parties, however unlike the real setting $\mathcal{Z}$ interacts with Sim instead of $\mathcal{A}$.
- $\text{HYB}^{\mathcal{F}_{\mathsf{sub}}}_{\Pi,\mathcal{A},\mathcal{Z}}$ is the output of $\mathcal{Z}$ in the "hybrid" setting, where the honest parties are conducting the real protocol Π, but can also interact with functionality $\mathcal{F}_{\mathsf{sub}}$.[6] In this setting, $\mathcal{Z}$ interacts with $\mathcal{A}$ in the same way as in the real world.

Lemmas 5 and 4 show that for various sets of corrupted parties, there is a negligible probability that the environment's outputs $\text{IDEAL}_{\mathcal{F}_{\mathsf{cert}},\mathsf{Sim},\mathcal{Z}}$ versus $\text{HYB}^{\mathcal{F}_{\mathsf{coin}}}_{\Pi_{\mathsf{cert}},\mathcal{A},\mathcal{Z}}$ will differ, where $\mathcal{F}_{\mathsf{coin}}$ is a standard coin-flipping ideal functionality defined in Appendix A. These lemmas formally capture the properties stated informally in Section 4.1.

5. The SUC model implicitly handles much of the boilerplate information in the standard UC model, especially in the following ways: The set of parties is fixed and public. Messages contain a public authenticated header containing the message's sender and receiver, and secret content. The adversary sees the public header, but cannot tamper with or forge the header and cannot see the private content. Because of this, SUC proofs can omit explicitly writing down the sender and receiver ID. Furthermore, each copy of $\mathcal{F}$ has a session ID and ignores messages sent with a different session ID, allowing omission of the session ID in SUC proofs. The environment $\mathcal{Z}$ communicates directly only with the adversary $\mathcal{A}$ and can write inputs and read outputs from all parties. For additional details, see [94].

6. This is slight abuse of notation; in the SUC model strictly speaking $\mathcal{F}_{\mathsf{sub}}$ is a sub-functionality of the full functionality $\mathcal{F}$, however we notate this $\mathcal{F}_{\mathsf{sub}}$ to remind ourselves that the only aspects of $\mathcal{F}$ the parties will interact with are those of $\mathcal{F}_{\mathsf{sub}}$.

Lemma 4 (Honest user, dishonest server and $(\tau-1)$ groups)**.** *For every efficient adversary $\mathcal{A}_c$ there exists a adversary Sim_c such that for every efficient environment $\mathcal{Z}$ which statically maliciously corrupts $\mathcal{S}$ and $d < \tau$ groups:*

$$\left|\Pr[\text{HYB}^{\mathcal{F}_{coin}}_{\Pi_{cert},\mathcal{A}_c,\mathcal{Z}}] - \Pr[\text{IDEAL}_{\mathcal{F}_{cert},\mathsf{Sim}_c,\mathcal{Z}}]\right|$$

is negligible if hash function H is modeled as a random oracle and answers at most Q_{ro} queries, H' is a secure key derivation function, $\mathcal{T}$ is a secure threshold signature scheme, PRF is a pseudorandom function, $(\mathsf{Enc}, \mathsf{Dec})$ is an IND$-CPA-secure random-key-robust encryption scheme, and DDH is hard in $E(\mathbb{F}_p)$.

Notice the implication of Lemma 4: even if d groups collude with $\mathcal{S}$, the user will correctly reject any signature shares held by fewer than $(\tau - d)$ parties with all but negligible probability; this is the best we can hope for for a threshold approval mechanism with d malicious participants.

The full proof of Lemma 4 is available in Appendix B. The intuition of the proof of this statement is the same as the proof of client privacy in [15]. This protocol works via the same technical mechanism as Apple's PSI scheme. In Apple's scheme, the clients send encrypted key shares that $\mathcal{S}$ can only recover if $P_j = \alpha \cdot H(e)$, where e is the PHF of the client's content. Here, an honest child safety group that holds element e_j will send encrypted signature shares that $\mathcal{S}$ can only recover if $P_j = \alpha \cdot H(e_j)$. In the proof, the simulator will send these encrypted shares on the honest groups' behalf. Along the way the simulator must recover $\mathcal{A}^H_{\mathcal{S}}$'s Cuckoo table; it does so by emulating the random oracle H. If $\mathcal{A}$ is able to provide a signature for any element that fewer than $(\tau - d)$ honest parties vouched for with non-negligible probability, then $\mathcal{A}$ has either broken the threshold signature scheme, the random-key-robust IND-CPA$ encryption scheme, or DDH.

We now turn to proving server privacy and security against a coalition of dishonest groups and the user. For an adversary $\mathcal{A}$ and simulator Sim, let $\mathscr{Z}_{\mathcal{A},\mathsf{Sim}}$ be the set of environments which take actions that either cause a semi-honest user to eventually output 1 with all but negligible probability. We will only prove privacy for these environments. It is possible that malicious child safety groups could learn information about honest groups' sets by refusing to sign particular elements, but in doing so they would cause verification to fail. The server could notice this in Step 5(a)ii and prevent those groups from participating in future rounds.

Lemma 5 (Honest server, dishonest $(N - \tau)$ groups and user)**.** *For every efficient adversary $\mathcal{A}_p$ there exists an efficient adversary Sim_p, such that for every efficient environment $\mathcal{Z} \in \mathscr{Z}_{\mathcal{A}_p,\mathsf{Sim}_p}$ which maliciously statically corrupts up to $(N - \tau)$ groups $\mathcal{G}_M$ and may semi-honestly corrupt the user $\mathcal{U}$:*

$$\left|\Pr[\text{HYB}^{\mathcal{F}_{coin}}_{\Pi_{cert},\mathcal{A}_p,\mathcal{Z}}] - \Pr[\text{IDEAL}_{\mathcal{F}_{cert},\mathsf{Sim}_p,\mathcal{Z}}]\right|$$

is negligible if Decision Diffie-Hellman (DDH) is hard in $E(\mathbb{F}_p)$, hash function H is modeled as a random oracle, and PRF is a pseudorandom function.

Observe that Lemma 5 captures our informal privacy definition: The ideal functionality will not reveal any information about X_i to the user, and so as long as an honest $\mathcal{U}$ would output 1 (i.e. all signatures verify), this property continues to hold in Π_{cert} even if all groups and the user are corrupt. Without this condition on the signature verification, privacy no longer holds; in other words, the child safety organizations could learn about each others' sets, but only by causing signature verification to fail.

The full proof of Lemma 5 is in Appendix B. The idea of the proof is that Sim_p simulates the pdata by building it out of only dummies and using a freshly sampled blinding key α'. This is indistinguishable from random by the same logic as the proof of server privacy in [15] under the assumption that DDH is hard in $E(\mathbb{F}_p)$ and H is a random oracle. As stated in the lemma statement, we only require this simulation to hold for environments in which a (semi-) honest user outputs 1.

4.3. Implementation and Benchmarks

Total communication overhead is $O(|X|N)$. We implemented Protocol Π_{cert} in Go using NIST P-256 and bilinear pairings on a 256-bit Barreto-Naehrig curve. The parties were run sequentially for $N = 3$ and $\tau = 2$ at the 128-bit security level on a 6-core Intel i7-10710U processor at 1.10GHz with a 12MB cache and 32GB RAM. For a matchlist containing 2^{20} elements, the server's runtime was 466s, the groups' average runtime was 255s, and the linear verification time was 469s. The verification time for a faster verification algorithm (see below) was 0.25s. The code is available at https://github.com/citp/pvphm.

Faster Public Verification. By running the full Π_{cert}, any user could get cryptographic assurance that each element of pdata was either a real hash of an image held by at least τ child safety groups, or a pseudorandom value. Verification of this fact naively requires checking a number of signatures equal to the size of the Cuckoo table (Step 6). To speed this process, after Π_{cert} has been run, child safety groups may run $\mathsf{TVerify}$ as members of the public, and if the result passes, output a share of a signature $\rho_i = \mathsf{TSign}(\mathsf{sk}_i, \mathsf{pdata})$. This then allows users $\in \mathcal{U}$ to verify $\mathsf{TVerify}(\mathsf{vk}, \mathsf{TAggSign}(\{\rho_i, \ldots, \rho_\tau\}, \mathsf{pdata}), \mathsf{pdata})$ by checking only one signature. Such verification takes only 0.25s on the same hardware. We still recommend that the signatures on each element be published as well, for users who wish to check the full set manually.

5. Proof of Non-Membership in the Hash Set

As discussed in Section 2.4, online services that deploy PHM could face immense pressure to include non-CSAM matches in the hash set. Apple promised that it "would refuse such demands" and that "this technology is limited to detecting CSAM stored in iCloud and we [Apple] will not accede to any government's request to expand it" [95].

Users may doubt these types of promises by online services and (potential) auditors. While Apple, for example,

has refused some government requests for data access—including a high-profile case involving the FBI [96]—it has also acceded to other requests. Apple now stores the data of Chinese users on servers physically located in China, with lesser encryption standards, and in data centers managed by a government-owned cloud computing company [58], [97].

Given these foreseeable risks of pressure and scope creep, it is in the interest of users and online services to be able to cryptographically prove that a PHM hash set does *not* contain specific images. This property mitigates—though does not eliminate—one of the concerns described in Section 2.4, that Apple's system could be used to censor or surveil sending of legitimate images. We achieve this property by constructing a zero-knowledge "proof of non-membership" in the hash set.

5.1. Goals and Threat Model

This protocol has two primary goals.

Limited exception to server privacy. The online service should be able to prove that the elements of a set of hashes for known legitimate images are *not* elements in the PHM hash set, as a deliberate relaxation of server privacy. Achieving this goal necessitates the server's participation in the protocol, or else the resulting protocol could violate server privacy more seriously: If clients could prove non-membership of hashes without the server's help, a client wishing to store CSAM could use the proof protocol to repeatedly test images with small perturbations until finding one with a hash distinct enough to avoid detection. It could also eventually allow enumeration of the server's private set.

Non-interactivity. The server should be able to publish the proof once, clients should be able to verify separately.

Soundness. The server should not be able to prove non-membership of a hash that is on the hash list.

5.2. Protocol for Proof of Non-Inclusion

Proving membership of an element in an otherwise-secret set is a common problem that can be solved directly with zero-knowledge proofs on commitments [98], [99], accumulators [100]–[102], or private set intersection [103], [104]. Proving non-membership is slightly trickier. A negative accumulator [105], [106] would allow proving non-membership in the set. However, we wish to make use of the pdata that is already published in the Apple PSI protocol.

Our approach is to combine a standard zero-knowledge proof of knowledge (ZKPoK) of a discrete log with a homomorphic commitment scheme. We take advantage of the fact that Apple's set is encoded in the form of a blinded Cuckoo table where an element e appears to the public as either $P_{h_1(e)} = \alpha \cdot H(e)$ or $P_{h_2(e)} = \alpha \cdot H(e)$, where α is Apple's secret blinding key. If the hash to be checked for non-membership is x, this will allow Apple to prove knowledge of α such that:

$$L = \alpha \cdot G \bigwedge P_{h_1(x)} \neq \alpha \cdot H(x) \bigwedge P_{h_2(x)} \neq \alpha \cdot H(x).$$

Protocol $\Pi_{\text{zk-pok-nm}}$

Shared parameters: Hash value x, size of Cuckoo table n, generator G of EC group $E(\mathbb{F}_q)$, domain universe $\mathcal{U}$, hash function $H : \mathcal{U} \to E(\mathbb{F}_q)$, Cuckoo table hash functions $h_1, h_2 : \mathcal{U} \to \{1, 2, \ldots, n\}$, pdata containing $L = \alpha \cdot G$ and $P_i = \alpha \cdot H(e_i)$—the blinded hash of element e_i at location i of the Cuckoo table—for $i \in \{1, \ldots, n\}$, homomorphic commitment scheme (HCom, HDecom, HAddCom)
Inputs: $\mathcal{P}$ holds secret blinding key α such that $P_i = \alpha \cdot H(e_i)$ for all locations $i \in \{1, \ldots, n\}$ in the Cuckoo table
Outputs: $\mathcal{V}$ accepts or rejects a proof produced by $\mathcal{P}$

1) $(\mathcal{P} \to \mathcal{V})$ Prover chooses $r, s, t \xleftarrow{\$} \mathbb{F}_q$, uses HCom to commit to α, r, s, and t and sends $R = r \cdot G, S = s \cdot H(x), T = t \cdot H(x)$ to $\mathcal{V}$.
2) $(\mathcal{V} \to \mathcal{P})$ $\mathcal{V}$ chooses i as follows: With probability 1/2, set i to 0, or with probability 1/6 each, set i to either 1, 2, or 3. Send i to the Prover.
3) $(\mathcal{P} \to \mathcal{V})$
   a) If $i = 0$, Prover decommits r, s, and t. $\mathcal{V}$ accepts if
   $$r \cdot G = R \bigwedge s \cdot H(x) = S \bigwedge t \cdot H(x) = T$$
   and if the commitment is valid, otherwise it rejects.
   b) If $i = 1$, $\mathcal{P}$ decommits $a' = \alpha + r$. $\mathcal{V}$ accepts if and only if $a' \cdot G = L + R$ and the commitment is valid.
   c) If $i = 2$, $\mathcal{P}$ decommits $a' = \alpha + s$. $\mathcal{V}$ accepts if and only if $a' \cdot H(x) \neq P_{h_1(x)} + S$ and the commitment is valid.
   d) If $i = 3$, $\mathcal{P}$ decommits $a' = \alpha + t$. $\mathcal{V}$ accepts if and only if $a' \cdot H(x) \neq P_{h_2(x)} + T$ and the commitment is valid.

Figure 3. $\Pi_{\text{zk-pok-nm}}$, a zero knowledge proof of non-membership of x in the Cuckoo table

We use a protocol for ZKPoK of a discrete log that yields a soundness error of approximately 1/6. Therefore, it must be repeated $128 \log_2(6) \approx 331$ times to achieve 128-bit security. We compute these proofs in parallel, and we make them non-interactive using the Fiat-Shamir heuristic.

We note that the efficiency and soundness of our protocol *do not depend on the size of Apple's Cuckoo table* and the protocol remains useful even if the hash set X grows large.

Proofs of Security. Proofs of soundness, knowledge soundness, and zero knowledge are standard and can be found in Appendix C.

5.3. Implementation and Benchmarks

We use the Pedersen scheme over NIST curve P-256 for homomorphic commitments [107]. The runtime of $\Pi_{\text{zk-pok-nm}}$ is presented in Table 1. We ran tests on a 6-core Intel i7-10710U processor at 1.10GHz with a 12MB cache and 32GB of RAM. The code is available at https://github.com/citp/pvphm.

Our single-threaded implementation requires 147 ms to generate a non-interactive proof and 66 ms to verify it at the 128-it security level, independent of the hash set size.

This makes our approach efficient enough to apply to databases of banned content, such as 220 political images CitizenLab found to be banned in WeChat [108] in less

Round	Time
Prover (Round 1)	897.7 μs
Verifier (Round 2)	2.409 μs
Prover (Round 3)	0.534 μs
Verifier (Verification)	441.9 μs
One iteration (Prover + Verifier)	1,342 μs
Prover (Non-Interactive)	147 ms
Verifier (Non-Interactive)	66 ms
Total (Interactive)	515 ms
Total (Non-Interactive)	**213 ms**

TABLE 1. RUNTIME OF $\Pi_{\text{ZK-POK-NM}}$ AT 128-BIT SECURITY

than a minute, or a criticized Russian database of banned "extremist" materials [109] in about 15 minutes.

6. Guaranteed Eventual Detection Notification

In the previous two sections, we described transparency methods that improve the trust in the hash set itself. In this section, we turn to a different question: how to improve transparency in the implementation.

A surprisingly underexamined question in the PHM debate is: what guarantees should be given to a user when their content has a match? In Apple's proposal, a positive match would be sent to a human moderator, who would "confirm that [the image was] CSAM material, in which case they would disable the offending account and refer the account to a child safety organization" [64]. It is not clear , however, what action Apple would take after a false positive (i.e., the threshold for reconstructing adkey was reached even though the user never stored CSAM).

This question has been considered more carefully in other areas of encryption policy. The Carnegie Endowment for International Peace developed a consensus report on encryption policy in 2019 in the context of law enforcement access to encrypted smartphones, and some of the principles neatly translate to PHM. One of the Carnegie principles is "Auditability: When a phone is accessed, the action is auditable to enable proper oversight, and is eventually made transparent to the user (even if in a delayed fashion due to the need for law enforcement secrecy)" [35]. Abelson et al. touch on this question for PHM systems, suggesting that they would "need to be designed in such a way that users could eventually learn which content was scanned ... and what was ultimately made available to the authorities" [20].

Informing users about false positive matches would be an important step forward for transparency and accountability. A user could also learn if they have been targeted with adversarially generated images that mimic CSAM hashes. For Apple's protocol, a user could learn when to rotate their adkey to prevent future disclosures.

Because differentiating between a true or false positive depends on subsequent manual review, whether by the online service, a child safety group, or a law enforcement agency, immediately notifying a user is infeasible. A delay is also important to prevent a malicious user from rapidly iterating images and testing for inclusion in the PHM system's hash set. Immediate notification could additionally tip off users who might become investigation targets.

In this section, we construct a delayed user notification system. When a user sends content that matches the online service's hash set, the service learns about the match immediately, and the user learns about the match after a delay.

6.1. Goals and Threat Model

Informally, protocol $\Pi_{\text{ev-notif}}$ has the following goals.

Privacy. Computationally bound users should learn nothing from the protocol until the delay has elapsed. After the delay is complete, a user should learn whether their own adkey was disclosed to the server and learn nothing about any other user's adkey.

Note that because our protocol involves a secure two-party computation (2PC), we must ensure that the very act of executing the protocol does not give away whether or not the server learned adkey until after delay. Thus, we design the protocol to be run once per timestep regardless of whether or not users have met the threshold of matches, and the online service will simply not learn anything if the user does not meet the threshold. For this reason, we would expect timesteps to be reasonably short (for instance, 1 day) to enable an online service to learn matching users' adkeys quickly. However, delay could be set to an arbitrarily longer period, for example months or many images shared.

Guaranteed notification. If a malicious server learns a user's adkey, then after delay has passed and if no party aborts, the user learns that this has happened.

Correctness. If the user acts as it would in the honest protocol, then the server learns the user's adkey immediately after timestep t ends. We do not attempt this property against a malicious user, since the Apple PSI protocol already fails correctness against a malicious user [15].

6.2. Protocol for Eventual Notification

We present protocol $\Pi_{\text{ev-notif}}$ for guaranteed eventual notification, which allows users to learn when their content was part of a positive match after a delay. The full description of the protocol is in Figure 4; we summarize it here.

We make one change to Apple's existing PSI protocol. Let adkey be the key the user uses to encrypt her associated data for Apple's threshold PSI scheme with associated data. In [15] the server learns secret shares of adkey each time the user sends a voucher containing a hash matching Apple's set. In our protocol, the user will still encrypt her data using adkey, but when the user sends vouchers, she will send shares of a new intermediate key ika instead of adkey. Our protocol requires the online service and the user to engage in a 2PC for the service to learn the final adkey, in such a way that the user will be able to, after a delay, learn whether or not the service learned the adkey from the 2PC.

In more detail, we break time up into epochs. At the start of epoch t, the server commits to encryption key ek_t, which can be the same across all users. Furthermore, during each timestep each user has a nonce nonce that is user and

Protocol $\Pi_{\mathsf{ev\text{-}notif}}$

Shared Parameters: Key length k, pseudorandom user-and-epoch-specific k-bit nonce nonce known to both parties, a timestep delay delay, encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$.
Inputs: At the beginning of time epoch t, the Server inputs encryption key ek_t in the keyspace of $(\mathsf{Enc}, \mathsf{Dec})$. At the end of epoch t, the server additionally inputs its guess ikb for the user's intermediate key. The user inputs its AEAD key adkey and intermediate key ika.
Commit Phase: At the beginning of timestep t, the server publishes a commitment c committing to ek_t.
Main Phase: At the end of timestep t, let ikb be the server's guess for the intermediate key. The server enters ek_t and ikb into a malicious-secure 2-party computation computing:

$$\mathsf{Enc}(\mathsf{ek}_t, \mathsf{nonce} \oplus (\mathsf{adkey} \wedge \mathbb{1}_{\mathsf{ika}=\mathsf{ikb}}))$$

where $\mathbb{1}_{\mathsf{ika}=\mathsf{ikb}}$ is all 1s if $\mathsf{ika} = \mathsf{ikb}$ below. In our implementation, Enc is the AES block cipher. The user inputs its true intermediate key ika and its adkey. The 2PC outputs its return statement to both parties.
Let the output of the 2PC be e. The server outputs $\mathsf{Dec}(\mathsf{ek}_t, e)$.
Post-delay Phase: After delay time has passed, the server opens the commitment c to reveal ek'_t. The user then outputs $\mathsf{Dec}(\mathsf{ek}'_t, e)$.

Figure 4. Protocol $\Pi_{\mathsf{ev\text{-}notif}}$ for guaranteed eventual notification

epoch specific. Both the user and the server can generate nonce from public information.

At the end of each timestep, the user and server conduct a two-party computation. During epoch t, the server may or may not have been able to reconstruct the shares of ika sent by the user. Either way, the server will input ikb into a 2PC as a "guess" for the intermediate key. If there are not enough shares to reconstruct, ikb is random. If there are enough shares to reconstruct, then ikb is the result of that reconstruction (which is equal to ika if the user is honest).

The output of the 2PC is a ciphertext e encrypted with ek_t. If $\mathsf{ikb} = \mathsf{ika}$, then e is an encryption of $\mathsf{adkey} \oplus \mathsf{nonce}$; if $\mathsf{ikb} \neq \mathsf{ika}$ then the encryption is of nonce. The server is able to immediately decrypt e using ek_t. However, the user must wait until the server opens the commitment to ek_t, which happens after a delay of delay many timesteps.

Proofs of Security. We prove the privacy of Protocol $\Pi_{\mathsf{ev\text{-}notif}}$ against a malicious server and user, and we prove that an honest user will always detect whether their adkey was learned by a malicious server or the server deviated from the protocol. We interpret server abort as malicious; the user should rotate her adkey in this case. Security proofs can be found in Appendix D.

6.3. Implementation and Benchmarks

We implemented the circuit capturing C in the open source Efficient Multi-Party Computation Toolkit (EMP-Toolkit) [110] using Authenticated Garbling and Efficient Maliciously-Secure 2-Party Computation [111], with AES-128 as the encryption scheme (since longer messages were not desired, a simple block cipher suffices for encryption). The circuit outputs $\mathsf{AES}(\mathsf{ek}_t, \mathsf{nonce} \oplus (\mathsf{adkey} \wedge \mathbb{1}_{\mathsf{ika}=\mathsf{ikb}}))$, where $\mathbb{1}_{\mathsf{ika}=\mathsf{ikb}}$ is all 1s if $\mathsf{ika} = \mathsf{ikb}$, and all 0s otherwise. We used the Bristol AES-128 "non-expanded" circuit [112] as our original AES circuit. Our $\mathsf{ev\text{-}notif}$ circuit contained 25509 XOR gates, 1693 NOT gates, and 7055 AND gates, of which 25124 XOR, 1692 NOT, and 6800 AND gates were part of the original AES circuit.

Averaged over 100 runs with random valid inputs, the protocol has an online time of 0.747 milliseconds and an offline preprocessing time of 38.2 milliseconds, not including network delays. The test machine had a 6-core Intel i7-10710U processor at 1.10GHz with a 12MB cache and 32GB of RAM. The code is available on GitHub at https://github.com/citp/pvphm.

7. Conclusion

In this work, we proposed protocols for public verification of PHM that would increase trust in the hash set and the implementation. The protocols would allow certification of the hash set by external groups, proof that particular content is not in the hash set, and eventual notification of false positives. We urge continued research and dialogue about how to implement content moderation in E2EE environments and improve trust in content moderation systems, including through both technical and nontechnical means.

Acknowledgements

Many thanks to Lior Rotem Benvenisty for providing an insight that improved Section 5. This work was supported by the Center for Information Technology Policy at Princeton University and a cryptography research grant from Ripple.

Appendix A: Coin-flipping functionality for Section 4

Our proofs are written in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model, meaning the parties have access to a shared coin flipping functionality (see Figure 5). The server and groups will call this functionality ℓ times to generate an ℓ-bit seed. This coin flipping protocol is standard and can be considered as a common random string shared among the groups and server. See [113], [114] for more on UC coin flipping.

Functionality $\mathcal{F}_{\mathsf{coin}}$

Coin flip:
1) Receive (Flip) from each group in $[N]$ and $\mathcal{S}$. Ignore repeat messages from a group until all groups have sent this message.
2) When all parties sent Flip, randomly choose $b \in \{0, 1\}$.
3) Send (Result, b) to all parties.

Figure 5. Standard ideal functionality $\mathcal{F}_{\mathsf{coin}}$ that allows $\mathcal{S}$ and N groups to flip a shared ideal coin

Appendix B: Security Proofs from Section 4

Proof of Lemma 4. Consider the simulator Sim_c which does the following:

1) Emulate random oracle $H : \mathcal{E} \to E(\mathbb{F}_p) \setminus \{\mathcal{O}\}$ for $\mathcal{A}_{\mathcal{S}}^H$ by doing the same procedure as in Sim_s of [15]. To summarize: Begin by initializing an empty list L_H. For each of a bounded number of queries on $e \in U$: check L_H to see if it contains a triple (e, β, R); if so, send it to the $\mathcal{A}_{\mathcal{S}}^H$. If not, then sample $\beta \leftarrow \mathbb{F}_q \setminus \{0\}$, set $R = \beta \cdot G$, add (e, β, R) to L_H, and send R to $\mathcal{A}_{\mathcal{S}}^H$.
2) Initialize a pair of efficient adversaries $\mathcal{A}_{\mathcal{S}}^H$ and $\mathcal{A}_{\mathcal{G}_M}^H$, both with access to random oracle H. Always forward communication normally between these machines and $\mathcal{Z}$, and abort of either machine aborts.
3) Run the key generation process $\mathsf{TKeygen}$ on behalf of the honest groups, interacting with $\mathcal{A}_{\mathcal{G}_M}^H$. Recombine the vk_i into vk as the honest user would, and store vk.
4) Pass along messages between all ideal and adversarial parties to conduct $\mathcal{F}_{\mathsf{coin}}$. Store the seed generated honestly from the output.
5) Eventually, the honest groups will output a message sending X_i to $\mathcal{F}_{\mathsf{cert}}$, and the malicious parties will output X_i to send to $\mathcal{S}$. Forward the malicious parties' sets to $\mathcal{A}_{\mathcal{S}}^H$, and send empty sets to $\mathcal{A}_{\mathcal{S}}^H$ on behalf of the honest parties. When $\mathcal{F}_{\mathsf{cert}}$ outputs the honest parties' X_i to send to $\mathcal{S}$, send the sets to $\mathcal{A}_{\mathcal{S}}^H$ as messages from the honest parties and also store the honest X_i for later.
6) Eventually, $\mathcal{A}_{\mathcal{S}}^H$ will output the pdata (consisting of $h_1, h_2, L, (P_1, \ldots, P_n)$). Send this to $\mathcal{A}_{\mathcal{G}_M}^H$ but also process it further in the next step.
7) Extract the set of "actual" elements X' and dummies D' stored in the pdata by doing the following. For each element (e, β, R) of L_H:
   a) Augment each tuple (e, β, R) into (e, β, R, P, j) where $P = \beta L$, and j is set to either $\perp$ (if there is no $P_{j'} = P$) or to j' if $P_{j'} = P$. There should be no repeats because the simulation already aborted if any $P_j = P_{j'}$ for $j \neq j'$.
   b) For each tuple, if $j \neq \perp$, let $D'[j] = e$ if there is a tuple of (e, β, R, P, j) such that $\mathsf{PRF}(\mathsf{seed}, j) = e$. Let $d_j = e$.
   c) For each tuple, if $j = h_1(e)$ or $j = h_2(e)$, then add e to X' as in [15].

   So each element of pdata corresponds either to an element X', an element of D', or neither. X' and D' put together create a Cuckoo table T'. If T' does not contain exactly n elements recovered in this way, halt. Pass h_1, h_2, T' to $\mathcal{F}_{\mathsf{cert}}$.
8) Eventually the honest groups send $(\mathsf{Vouch}, j, V_{i,j})$ to $\mathcal{F}_{\mathsf{cert}}$. Use the stored X_i from honest groups from step 5 and use the pdata to honestly build sets $E_{i,j}$ using the input set $X_i \cup \{\mathsf{PRF}(\mathsf{seed}, j)\}$.
9) Send the simulated $E_{i,j}$ to $\mathcal{A}_{\mathcal{S}}^H$ along with the adversarial $E_{i,j}$ sent by $\mathcal{A}_{\mathcal{G}_M}^H$.
10) $\mathcal{A}_{\mathcal{S}}^H$ will eventually send aggregate signatures $\sigma_{*,j}$. For each $j \in [n]$, let $s_j = \mathsf{TVerify}(\mathsf{vk}, \sigma_{*,j}, P_j)$.
11) For all $j \in [n]$, send $(\mathsf{Vouch}, j, \{T'[j]\})$ to $\mathcal{F}_{\mathsf{cert}}$ on behalf of all malicious groups $\mathcal{G}_i$ in $\mathcal{G}_M$.
12) Send $(\mathsf{PublishSig}, j, s_j)$ to $\mathcal{F}_{\mathsf{cert}}$ on behalf of $\mathcal{S}$.
13) $\mathcal{F}_{\mathsf{cert}}$ eventually sends $(\mathsf{Verified}, j, b_j)$ to $\mathcal{U}$ for each j.
14) The ideal user outputs 1 if all $b_j = 1$, 0 otherwise.

We must show both that the simulator does not fail and that $\mathcal{Z}$'s interaction with this simulator is indistinguishable from its interaction with $\mathcal{A}_c$.

First, consider the ways the simulator might fail. The simulator will halt at step 6 if any P'_j value is equal to another, which could happen if:

1) a pseudorandom dummy collided with either another dummy or a real element of X, causing the ideal functionality to abort
2) there is a pair of elements $e \neq e' \in T$ (dummy or real) for which $H(e) = H(e')$, or

and additionally will halt at step 7 if

3) the extraction of X' or D' fails (and thus $|T'| < n$) because $\mathcal{A}_{\mathcal{S}}^H$ guessed some value $y = H(e)$ from e without querying H.

We expect the chance that n unique inputs to the PRF have a collision is at most $\epsilon_{\mathsf{coll},n,2^\ell}$ plus the negligible chance ϵ_{prf} with which the PRF can be distinguished from random, so we can upper bound the chance of failure 1 above by $\epsilon_{\mathsf{coll},n,2^\ell} + \epsilon_{\mathsf{prf}}$.

For the next component, 2, the chance that some pair of unequal elements $e \neq e' \in T$ yield the same hash $H(e) = H(e')$ for random oracle H is exactly $\epsilon_{\mathsf{coll},n,q}$.

Finally, for 3, Sim_c will fail if $\mathcal{A}_{\mathcal{S}}^H$ guessed some value $y = H(e)$ from e without querying H, or by querying $H(e)$ after y was chosen. This chance is at most $(n+\mathsf{Q}_{\mathsf{ro}})/(q-1)$ where Q_{ro} is the number of queries made to H.

Thus, the total chance the simulation fails is at most $\epsilon_{\mathsf{coll},n,2^\ell} + \epsilon_{\mathsf{coll},n,q} + \frac{n+\mathsf{Q}_{\mathsf{ro}}}{q-1} + \epsilon_{\mathsf{prf}}$.

Now we turn to $\mathcal{Z}$'s ability to distinguish the ideal from the hybrid world. The view of the environment consists of the user's output in addition to the views of $\mathcal{A}_{\mathcal{S}}^H$ and $\mathcal{A}_{\mathcal{G}_M}^H$:

- The $\mathsf{TKeygen}$ exchanges between the honest and malicious groups in step 3
- The $\mathcal{F}_{\mathsf{coin}}$ exchanges in step 4
- The X_i messages to $\mathcal{A}_{\mathcal{S}}^H$ sent in step 5
- The pdata given to $\mathcal{A}_{\mathcal{G}_M}^H$ in step 6
- The $E_{i,j}$ sent to $\mathcal{S}$ from both the honest and malicious groups in step 9
- The user output 0 or 1

Notice that all of these items except the user output have the same distribution in both the real and ideal worlds: For $\mathsf{TKeygen}$, Sim_c acts exactly as the real honest parties would, the $\mathcal{F}_{\mathsf{coin}}$ calls are the same in both worlds, and the X_i sent by Sim_c to $\mathcal{A}_{\mathcal{S}}^H$ are, for honest parties, the same X_i sent by $\mathcal{F}_{\mathsf{cert}}$ to Sim_c. The pdata are generated by $\mathcal{A}_{\mathcal{S}}^H$ and so therefore do not provide any new information to $\mathcal{Z}$ when passed to $\mathcal{A}_{\mathcal{G}_M}^H$. The $E_{i,j}$ are generated by Sim_c in exactly the way the honest parties would generate them when running the protocol.

The only remaining task is to show that the user output is indistinguishable in the hybrid and ideal worlds. In the

hybrid world, the user will output 1 if and only if for each signature $\sigma_{*,j}$ obtained in step 10, $\mathsf{TVerify}(\mathsf{vk}, \sigma_{*,j}, P_j)$ outputs 1. In the ideal world, the user will output 1 if and only if each b_j value sent in step 13 are 1. For a given b_j to be 1 in the ideal world, it must be true that $T'[j]$ appeared in at least τ sets $V_{i,j}$ across steps 8 and 11, and $s_j = \mathsf{TVerify}(\mathsf{vk}, \sigma_{*,j}, P_j) = 1$.

There are two cases in which the ideal b_j and the real signature verification could diverge: either $b_j = 1$ but $\mathsf{TVerify}(\mathsf{vk}, \sigma_{*,j}, P_j) = 0$, or $b_j = 0$ but $\mathsf{TVerify}(\mathsf{vk}, \sigma_{*,j}, P_j) = 1$. The first case is impossible because $b_j = 1$ is only true if $s_j = 1$, and Sim_c set s_j as the output of $\mathsf{TVerify}$. So we are left with the other case, where $b_j = 0$ but $\mathsf{TVerify}(\mathsf{vk}, \sigma_{*,j}, P_j) = 1$.

In this case, $s_j = 1$ since it was defined as the output of $\mathsf{TVerify}$. Thus the only way $b_j = 0$ is if fewer than τ groups included $T'[j]$ in their Vouch sets. By construction all d malicious parties vouch for $T'[j]$ in step 11, Thus fewer than $(\tau - d)$ honest groups $\mathcal{G}_i$ had $T'[j] \in X_i$. If $\mathcal{A}_{\mathcal{S}}^H$ is able to produce the valid signature $\sigma_{*,j}$ despite this, then either

- $\mathcal{A}_{\mathcal{S}}^H$ forged the threshold signature $\sigma_{*,j}$ despite having fewer than τ legitimate shares,
- $\mathcal{A}_{\mathcal{S}}^H$ managed to decrypt a ciphertext in $E_{i,j}$ despite not having the key,
- or $\mathcal{A}_{\mathcal{S}}^H$ successfully guessed the key $S_{i,j}$ from $Q_{i,j}$ even though $P_j \neq \alpha \cdot H(T'[j])$.

All three of these possibilities occur with at most negligible probability, combining them with a union bound we see that $\mathcal{Z}$ can distinguish with probability at most $\epsilon_{\mathsf{ts}} + \epsilon_{\mathsf{enc}} + 2\epsilon_{\mathsf{orig}}$.

Thus, the overall probability with which either the simulation will fail or $\mathcal{Z}$ will distinguish is at most

$$\epsilon_{\mathsf{coll},n,2^\ell} + \epsilon_{\mathsf{coll},n,q-1} + \frac{n + \mathsf{Q}_{\mathsf{ro}}}{q-1} + \epsilon_{\mathsf{prf}} + \epsilon_{\mathsf{ts}} + \epsilon_{\mathsf{enc}} + 2\epsilon_{orig}.$$

This is negligible in a security parameter for an appropriate choice of ℓ, n, q, and Q_{ro}, and so this proves the statement. □

Proof of Lemma 5. Consider the following simulator Sim_p, interacting with the ideal $\mathcal{S}$, honest parties $\mathcal{G}_O$, possibly an honest user $\mathcal{U}$, and the environment $\mathcal{Z}$:

1) Emulate random oracle H as in [15].
2) Let $\mathcal{A}_{\mathcal{G}_M}^H$ be the portion of the efficient adversary $\mathcal{A}$ that acts as the malicious groups $\mathcal{G}_M$, exchanging messages with environment $\mathcal{Z}$, user $\mathcal{U}$, server $\mathcal{S}$, and honest parties $\mathcal{G}_O$. If the user is corrupt, let $\mathcal{A}_{\mathcal{U}}^H$ be the portion of $\mathcal{A}$ that acts as $\mathcal{U}$, exchanging messages with $\mathcal{Z}$, $\mathcal{G}_M$, $\mathcal{G}_O$, and $\mathcal{S}$. Always forward messages from these adversaries to and from the environment. Always abort if these machines ever abort.
3) As the groups in $\mathcal{A}_{\mathcal{G}_M}^H$ run $\mathsf{TKeygen}$, Sim_p acts on behalf of each honest group in $\mathcal{G}_O$ when running $\mathsf{TKeygen}$, sending and receiving messages from malicious groups in $\mathcal{G}_M$ as needed. Eventually the groups in $\mathcal{G}_M$ output verification key shares vk_i; Sim_p does the same on behalf of the honest groups $\mathcal{G}_O$.
4) If $\mathcal{U}$ is corrupt, Sim_p computes $\mathsf{vk} \leftarrow \mathsf{TKeyCombine}(\mathsf{vk}_1, \ldots, \mathsf{vk}_N)$ and sends it to $\mathcal{A}_{\mathcal{U}}^H$.
5) Sim_p relays messages between honest and corrupted parties and the $\mathcal{F}_{\mathsf{coin}}$ functionality as normal, and builds the ℓ-bit seed from the responses to $\mathcal{A}_{\mathcal{G}_M}^H$ as an honest party would.
6) Eventually the honest groups $\mathcal{G}_O$ output X_i to send to the functionality; Sim_p forwards these. $\mathcal{A}_{\mathcal{G}_M}^H$ outputs a message X_i to send to $\mathcal{S}$ for each malicious group $\mathcal{G}_i$. Sim_p ignores these and instead sends the empty set to $\mathcal{F}_{\mathsf{cert}}$ on behalf of each malicious group. Sim_p forwards all sets from $\mathcal{F}_{\mathsf{cert}}$ to $\mathcal{S}$.
7) The honest server eventually sends T of length n to the functionality, along with hash functions h_1 and h_2; T should contain only dummies and the elements input by at least τ honest groups.
8) The functionality sends to all groups n, h_1, and h_2. Sim_p picks random alternate blinding key $\alpha' \xleftarrow{\$} \mathbb{F}_q$. It computes $L' = \alpha' \cdot G$ and $P'_j = \alpha' \cdot H(d_j)$, where $d_j = \mathsf{PRF}(\mathsf{seed}, j)$ is the dummy element for all $j \in [n]$. If any of the P'_j values are equal, abort. Sim_p sends pdata' of $h_1, h_2, L', P'_1, \ldots, P'_n$ to the corrupted party adversary $\mathcal{A}_{\mathcal{G}_M}^H$ and, if the user is corrupted, to $\mathcal{A}_{\mathcal{U}}^H$.
9) The honest groups $\mathcal{G}_i \in \mathcal{G}_O$ eventually output $(\mathsf{Vouch}, j, V_{i,j})$; because these groups are honest $V_{i,j}$ will include d_j. Sim_p forwards these messages to $\mathcal{F}_{\mathsf{cert}}$.
10) $\mathcal{A}_{\mathcal{G}_M}^H$ eventually outputs $E_{i,j}$ for all j for malicious groups $i \in \mathcal{G}_M$.
11) Sim_p attempts decryption of the $E_{i,j}$. If there is no $E_{i,j}$ element that decrypted to a valid share of P'_j then Sim_p sends $(\mathsf{Vouch}, j, \{\})$ to the ideal $\mathcal{S}$ on behalf of malicious group $\mathcal{G}_i$. If there is a valid share, Sim_p sends $(\mathsf{Vouch}, j, \{d_j\})$ to the ideal $\mathcal{S}$ on behalf of $\mathcal{G}_i$.
12) The ideal $\mathcal{S}$ sends $(\mathsf{PublishSig}, j, 1)$ to $\mathcal{F}_{\mathsf{cert}}$ for all j.
13) By construction, and because $\mathcal{S}$ was honest, the two properties in Step 5 of $\mathcal{F}_{\mathsf{cert}}$ will have been met, and so $\mathcal{F}_{\mathsf{cert}}$ will output $(\mathsf{Verified}, j, 1)$ to $\mathcal{U}$ for all $j \in [n]$. If $\mathcal{U}$ is honest, this completes the simulation.
14) If the user is corrupt, Sim_p builds $\sigma_{*,j}$ values to send to the corrupt $\mathcal{A}_{\mathcal{U}}^H$ using the τ signature shares from the honest parties it conducted $\mathsf{TKeygen}$ on behalf of; input from the malicious groups is not needed. It uses $\mathsf{TCombine}$ to combine these into valid signature shares and sends them to $\mathcal{A}_{\mathcal{U}}^H$.

As with the proof of Lemma 4, we must show that the chance the simulation fails, and the chance $\mathcal{Z}$ can distinguish the real from the hybrid world, is negligible.

For the chance of failure, this simulation Sim_c will fail under either of the first two conditions as Sim_c from Lemma 4, which together bound the chance of failure by $\epsilon_{\mathsf{coll},n,2^\ell} + \epsilon_{\mathsf{coll},n,q-1} + \epsilon_{\mathsf{prf}}$.

Now we have to show that if the simulation does not fail, $\mathcal{Z}$ cannot distinguish between the ideal world where it interacts with Sim_p and the hybrid setting where it is interacting with $\mathcal{A}_p$. The view of $\mathcal{Z}$ contains the output of the semi-honest or honest $\mathcal{U}$ at the end of the protocol (by assumption, always 1), as well as the views of $\mathcal{A}_{\mathcal{G}_M}$ and $\mathcal{A}_{\mathcal{U}}$. These contain:

- Messages sent during $\mathsf{TKeygen}$ in step 3

- The outputs from calls to $\mathcal{F}_{\mathsf{coin}}$ in step 5
- The simulated pdata' or real $h_1, h_2, L, P_1, \ldots, P_n$ in step 8
- If the user is corrupt, then the signatures in step 14

The $\mathsf{TKeygen}$ and $\mathcal{F}_{\mathsf{coin}}$ messages have identical distributions in the real and ideal worlds, and so we are left with simulation failure if either the pdata simulation fails, or if the signatures allow distinguishing the real and ideal worlds.

First, the pdata: h_1 and h_2 are generated the same way in the ideal and real worlds. Similarly, L and L' have identical distributions. Since n is input by the honest $\mathcal{S}$ and therefore the same in both worlds, it only remains to show that the set of P_j values is indistinguishable from the ideal P'_j. Like in [15], the real $(L, P_1, \ldots, P_n)$ can be distinguished from random elements in $E(\mathbb{F}_p)$ under the Decision Diffie Hellman (DDH) assumption when H is a random oracle (even with knowledge of the element e_j such that $P_j = \alpha \cdot H(e_j)$) with at most negligible probability ϵ_{orig}. By the same reasoning, $(L', P'_1, \ldots, P'_n)$ is also indistinguishable from random with the same bound. Thus, the real pdata is indistinguishable from the simulated pdata' with probability at most $2\epsilon_{\mathsf{orig}}$.

Finally, the signatures. In the ideal world, the functionality will send $(\mathsf{Verified}, j, 1)$ to the user for all j, because T' contains only dummies and elements provided by at least τ honest parties (and therefore also vouched for by τ honest parties. In the real world, we conditioned on the fact that the honest user output 1, so the server must have sent enough valid signature shares to pass verification for each j.

Thus, the probability that $\mathcal{Z}$ can distinguish between the real and ideal worlds (or the simulator fails) is at most

$$\epsilon_{\mathsf{coll},n,2^\ell} + \epsilon_{\mathsf{coll},n,q-1} + \epsilon_{\mathsf{prf}} + 2\epsilon_{\mathsf{orig}}$$

which is negligible in a security parameter for appropriate choices of n, q, and ℓ. This completes the proof. □

Appendix C: Security Proofs from Section 5

Lemma 6 (Soundness)**.** $\Pi_{\mathsf{zk\text{-}pok\text{-}nm}}$ *has soundness of* $(1/6 - \delta)$*, where* δ *is the chance the prover breaks binding.*

Proof. This is a straightforward case analysis. Let δ be the chance the prover breaks binding on the commitment.

First, suppose Apple's input α is such that $L \neq \alpha \cdot G$. If $R = r \cdot G$ is set honestly, then either the commitment failed or $\mathcal{V}$ will reject if $i = 1$ (probability 1/6). If R is not honest, then $\mathcal{V}$ will reject if $i = 0$ (probability 1/2).

Next, suppose $P_{h_1(x)} \neq \alpha \cdot H(x)$. If $S = s \cdot H(x)$ is set honestly, then either the commitment failed (probability δ) or $\mathcal{V}$ will reject if $i = 2$ (probability 1/6). If S is not honest, then $\mathcal{V}$ will reject if $i = 0$ (probability 1/2).

Finally, suppose $P_{h_2(x)} \neq \alpha \cdot H(x)$. Similar to the previous case, if $T = t \cdot H(x)$ is set honestly, then either the commitment failed (probability δ) or $\mathcal{V}$ will reject if $i = 3$ (probability 1/6). If T is not honest, then $\mathcal{V}$ will reject if $i = 0$ (probability 1/2).

In all cases the verifier will reject with probability at least $(1/6 - \delta)$. □

Lemma 7 (Knowledge soundness)**.** $\Pi_{\mathsf{zk\text{-}pok\text{-}nm}}$ *has knowledge soundness of at least 1/2.*

Proof. Let $\mathcal{P}^\star$ be a (possibly cheating) prover which convinces an honest verifier it is truthful with probability $1/2 + \epsilon$. (A party that doesn't know a witness may succeed with probability 1/2 by following the protocol and hoping $i = 0$.) Observe that any cheating strategy which allows $\mathcal{P}^\star$ to successfully cheat when sent at least one of $i = 1$, $i = 2$, or $i = 3$, can be modified to a cheating strategy where it cheats on all three, using a similar method. We consider only these cheating provers for simplicity. Let X be an extractor which runs the protocol twice, rewinding through step 2 to send two choices of i. If exactly one of these two choices for i is 0 (which occurs with probability 1/2), then the extractor learns α by subtracting the values from the two iterations of round 3. This extractor succeeds with probability at least $\epsilon' \geq (\frac{1}{2} + \epsilon)^2 - (\frac{1}{2} + \epsilon)/2 = \frac{\epsilon}{2} + \epsilon^2$ by the Rewinding Lemma [115, Section 19.2]. □

Lemma 8 (Zero knowledge)**.** $\Pi_{\mathsf{zk\text{-}pok\text{-}nm}}$ *is statistically zero knowledge in the commitment-hybrid model (in which commitments are perfectly hiding and binding).*

Proof. Let Sim be a simulator which does the following:

1) Randomly pick $\hat{i}$ from the same distribution as i.
2) Set variables depending on the choice of $\hat{i}$:
   a) If $\hat{i} = 0$, randomly (honestly) pick r, s, and t. Pick a random value for $\hat{\alpha}$. Commit to all of these with HCom, and send honest $R = r \cdot G$, $S = s \cdot H(x)$, and $T = t \cdot H(x)$.
   b) If $\hat{i} = 1$, randomly pick $\hat{a'}$. Randomly pick $\hat{\alpha}$ and $\hat{r}$ such that $\hat{a'} = \hat{\alpha} + \hat{r}$. Randomly pick s and t. Commit to all these with HCom. Let $\hat{A} = \hat{a'} \cdot G$, and let $\hat{R} = \hat{A} - L$. Let $S = s \cdot H(x)$ and $T = t \cdot H(x)$ as normal. Send $\hat{R}$, S, and T.
   c) If $\hat{i} = 2$, randomly pick $\hat{s}$. Randomly pick $\hat{\alpha}$ conditioned on $(\hat{\alpha} + \hat{s}) \cdot H(x) \neq P_{h_1(x)} + \hat{s} \cdot H(x)$. Honestly pick r and t. Commit to $\hat{\alpha}$, $\hat{s}$, r, and t and send $R = r \cdot G$, $\hat{S} = \hat{s} \cdot H(x)$, and $T = t \cdot H(x)$.
   d) If $\hat{i} = 3$, randomly pick $\hat{t}$. Randomly pick $\hat{\alpha}$ conditioned on $(\hat{\alpha} + \hat{t}) \cdot H \neq P_{h_2(x)} + \hat{t} \cdot H(x)$. Honestly pick r and s. Commit to $\hat{\alpha}$, $\hat{t}$, r, and s and send $R = r \cdot G$, $S = s \cdot H(x)$, and $\hat{T} = t \cdot H(x)$.
3) Receive i from the verifier. If $\hat{i} = i$, continue. Else, rewind to step 1.
4) To simulate round 3, decommit honestly.

Observe that in all decommitted values a' are uniformly distributed, as in the real protocol. Also observe that $\hat{R}$ has the same (uniform) distribution as the honest R, and similarly for $\hat{S}$ and $\hat{T}$. The only remaining difficulty is in ensuring that the simulator terminates. No matter what the strategy $\mathcal{V}^*$ uses to pick i, it takes at most 6 expected repetitions to reach a repetition where $\hat{i} = i$; the simulator is thus polynomial time. □

Appendix D: Security Proofs from Section 6

To show privacy against the malicious server, we show that the server has a negligible chance of winning the following game $G^{\mathcal{S}}_{\mathsf{ev\text{-}notif}}$ with a challenger: The server chooses nonce and sends it to the challenger. The challenger chooses a random $(\mathsf{ika}, \mathsf{adkey})$ and plays the role of the user in Protocol $\Pi_{\mathsf{ev\text{-}notif}}$ (including playing the role of the ideal functionality of the 2PC); then the challenger presents either $(\mathsf{ika}, \mathsf{adkey})$ or freshly randomly chosen values from the same space. The server wins if they guess correctly whether the challenger presented the real $(\mathsf{ika}, \mathsf{adkey})$ chosen at the start, or fresh random values.

Lemma 9 (Privacy and detection against malicious server)**.**

a. If there is no match, then under the same assumptions as the 2PC protocol, the chance a malicious server wins game $G^{\mathcal{S}}_{\mathsf{ev\text{-}notif}}$ is negligible.

b. If the server is malicious and computationally bounded, protocol $\Pi_{\mathsf{ev\text{-}notif}}$ reveals to the client whether there was a match or allows the client to detect malicious activity at most delay periods later, assuming the commitment scheme is computationally binding, $(\mathsf{Enc}, \mathsf{Dec})$ is a computationally-binding committing encryption scheme, and the 2PC is malicious-secure up to abort.

Proof. To show (a), consider an execution of game $G^{\mathcal{S}}_{\mathsf{ev\text{-}notif}}$. The server chooses the nonce nonce and sends it to the challenger, and the challenger then chooses $(\mathsf{ika}, \mathsf{adkey})$ randomly. The malicious server sends a commitment to the challenger, then inputs its $(\mathsf{ek}_t, \mathsf{ikb})$ for the 2PC. The challenger checks whether $\mathsf{ika} = \mathsf{ikb}$; if so it returns $e = \mathsf{Enc}(\mathsf{ek}_t, \mathsf{adkey} \oplus \mathsf{nonce})$. If not, it returns $e = \mathsf{Enc}(\mathsf{ek}_t, \mathsf{nonce})$. The server sends an arbitrary commitment opening to the challenger. The challenger then flips a coin b. If $b = 0$, it reveals $(\mathsf{ika}, \mathsf{adkey})$. If $b = 1$, it rolls new random choices $(\mathsf{ika}', \mathsf{adkey}')$ and sends those instead. The adversarial server wins if it guesses b correctly.

The only information learned by the server is the output of the 2PC. It is clear from inspection that the only way for the server to get any information about either ika or adkey is if they manage to guess $\mathsf{ikb} = \mathsf{ika}$, or if they break the security of the 2PC. However, since ika was randomly chosen, the chance of choosing $\mathsf{ikb} = \mathsf{ika}$ is negligible in the security parameter of the encryption scheme for which ika is a key. Thus, (a) is true.

Consider (b). Suppose for a moment the commitment scheme is perfectly binding and the encryption scheme commits perfectly to the key and message. (That is, decryption of a ciphertext e will fail if any key/message combination other than the one used to encrypt e was used.)

Then, if no parties abort and the decommitment and the decryption during the post-delay phase delay time after timestep t do not fail, then the message m the user obtains from decryption must have been the same as the message obtained by the server, which is the only information learned by the server. If that message was nonce, then the client knows no information about adkey was leaked to the server during timestep, assuming the 2PC is secure. If the message was $\mathsf{nonce} \oplus \mathsf{adkey}$, the user knows this was also learned by the server during timestep t.

The chance of a failure in binding of the committing encryption or the commitment scheme, or a failure of security in the 2PC, are all negligible. So by a union bound on these failure rates, the user detects notification with all but negligible probability as long as no party aborts.

If the server aborts before the post-delay phase, the user also interprets this as a malicious action, and so the property continues to hold. □

Lemma 10 (Privacy against malicious client)**.** *Protocol $\Pi_{\mathsf{ev\text{-}notif}}$ reveals no information about the result of the match to a computationally bound malicious client, until after delay time periods, assuming $(\mathsf{Enc}, \mathsf{Dec})$ is IND$-CPA secure, the commitment scheme is computationally hiding, and the 2PC is malicious-secure.*

Proof. The only inputs given by the client are into the malicious-secure 2PC. So consider the other possible sources of information for the client: First, we note that the commitment scheme has at most a negligible chance of leaking information to the user about ek_t. Second, if the commitment scheme were perfectly hiding, we note that the encryption e has at most a negligible chance of leaking information to the user. Third, assuming the encryption holds, the only additional information the user could learn is from the 2PC. Thus, assuming the 2PC is secure, the encryption scheme is IND$-CPA-secure, and the commitment scheme is hiding, the semi-honest client learns no information about the result of a match. □

References

[1] T. Perrin and M. Marlinspike, *The Double Ratchet Algorithm*, 11 2016. [Online]. Available: https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf

[2] G. Kent, *Why does WhatsApp Enable End-to-End Encryption?*, 5 2018. [Online]. Available: https://about.fb.com/news/2018/05/end-to-end-encryption/

[3] Apple, *iMessage Security Overview*, 2 2021. [Online]. Available: https://support.apple.com/guide/security/imessage-security-overview-secd9764312f/1/web/1

[4] Google, *Messages End-to-End Encryption Overview*, 2 2022. [Online]. Available: https://www.gstatic.com/messages/papers/messages_e2ee.pdf

[5] E. Bursztein, E. Clarke, M. DeLaune, D. M. Elifff, N. Hsu, L. Olson, J. Shehan, M. Thakur, K. Thomas, and T. Bright, "Rethinking the detection of child sexual abuse imagery on the internet," in *The world wide web conference*, 2019, pp. 2601–2607.

[6] *PhotoDNA*, Microsoft, 2021. [Online]. Available: https://www.microsoft.com/en-us/photodna

[7] *Fighting child sexual abuse online*, Google, 2021. [Online]. Available: https://protectingchildren.google#tools-to-fight-csam

[8] *Resource Guide*, Global Internet Forum to Counter Terrorism. [Online]. Available: https://gifct.org/resource-guide/

[9] T. Llanos, "Current approaches to terrorist and violent extremist content among the global top 50 online content-sharing services," Organization for Economic Co-Operation and Development, Tech. Rep., August 2020. [Online]. Available: https://doi.org/10.1787/68058b95-en

[10] B. Goetting, "Google drive cloud storage service employs hash matching to detect pirated content," *Hot Hardware*, February 2017. [Online]. Available: https://hothardware.com/news/google-drive-cloud-storage-service-employs-hash-matching-to-detect-pirated-content

[11] O. Solon, "Facebook asks users for nude photos in project to combat 'revenge porn'," *The Guardian*, 11 2017. [Online]. Available: https://www.theguardian.com/technology/2017/nov/07/facebook-revenge-porn-nude-photos

[12] A. P. Namanya, I. U. Awan, J. P. Disso, and M. Younas, "Similarity hash based scoring of portable executable files for efficient malware detection in iot," *Future Generation Computer Systems*, 2020.

[13] I. W. Foundation, "Image hash list." [Online]. Available: https://www.iwf.org.uk/our-technology/our-services/image-hash-list/

[14] A. Kulshrestha and J. R. Mayer, "Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation," in *USENIX Security 2021*, 2021. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/kulshrestha

[15] A. Bhowmick, D. Boneh, S. Myers, K. Talwar, and K. Tarbe, "The apple psi system," Apple, Inc., Tech. Rep., 2021. [Online]. Available: https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf

[16] Apple, "Expanded protections for children," aug 5, 2021 version. [Online]. Available: https://web.archive.org/web/20210805200549/https://www.apple.com/child-safety/

[17] Public Safety Canada, "International statement: End-to-end encryption and public safety," 10 2020. [Online]. Available: https://www.esafety.gov.au/newsroom/blogs/end-end-encryption-challenging-quest-for-balance

[18] WeProtect Global Alliance, "A welcome step: a statement of support for apple's plans to detect child sexual abuse images," 9 2021. [Online]. Available: https://www.weprotect.org/library/a-welcome-step-a-statement-of-support-for-apples-plans-to-detect-child-sexual-abuse-images/

[19] Safety Tech Innovation Network, "Safety tech challenge fund," 2022. [Online]. Available: https://www.safetytechnetwork.org.uk/innovation-challenges/safety-tech-challenge-fund/

[20] H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, J. Callas, W. Diffie, S. Landau, P. G. Neumann, R. L. Rivest *et al.*, "Bugs in our pockets: The risks of client-side scanning," *arXiv preprint arXiv:2110.07450*, 2021.

[21] S. B. Franklin and G. Nojeim, "International coalition calls on apple to abandon plan to build surveillance capabilities into iphones, ipads, and other products," 8 2021. [Online]. Available: https://cdt.org/insights/international-coalition-calls-on-apple-to-abandon-plan-to-build-surveillance-capabilities-into-iphones-ipads-and-other-products/

[22] D. K. Gillmor, "Apple's new 'child safety' plan for iphones isn't so safe," *American Civil Liberties Union*, 8 2021. [Online]. Available: https://www.aclu.org/news/privacy-technology/apples-new-child-safety-plan-for-iphones-isnt-so-safe/

[23] M. D. Green and A. Stamos, "Apple wants to protect children. but it's creating serious privacy risks." *New York Times*, 8 2021. [Online]. Available: https://www.nytimes.com/2021/08/11/opinion/apple-iphones-privacy.html

[24] S. Kamara, M. Knodel, E. Llansó, G. Nojeim, L. Qin, D. Thakur, and C. Vogus, "Outside looking in," *Center for Democracy and Technology*, 2021. [Online]. Available: https://cdt.org/wp-content/uploads/2021/08/CDT-Outside-Looking-In-Approaches-to-Content-Moderation-in-End-to-End-Encrypted-Systems.pdf

[25] J. Sanchez, "Apple's iphone: Now with built-in surveillance," *Cato Institute*, 8 2021. [Online]. Available: https://www.cato.org/blog/apples-iphone-now-built-surveillance

[26] I. McKinney and E. Portnoy, "Apple's plan to "think different" about encryption opens a backdoor to your private life," *Electronic Frontier Foundation*, 8 2021. [Online]. Available: https://www.eff.org/deeplinks/2021/08/apples-plan-think-different-about-encryption-opens-backdoor-your-private-life

[27] N. Meysenburg, L. Sarkesian, R. Schulman, and A. W. Thompson, "A technical explainer on apple's concerning privacy changes," *New America*, 8 2021. [Online]. Available: https://www.newamerica.org/oti/briefs/a-technical-explainer-on-apples-concerning-privacy-changes/

[28] N. Patel, R. Pfefferkorn, and J. King, "Here's why apple's new child safety features are so controversial," 8 2021. [Online]. Available: https://www.theverge.com/22617554/apple-csam-child-safety-features-jen-king-riana-pfefferkorn-interview-decoder

[29] P. Rosenzweig, "The law and policy of client-side scanning," *Lawfare*, 8 2020. [Online]. Available: https://www.lawfareblog.com/law-and-policy-client-side-scanning

[30] R. Canetti and G. Kaptchuk, "The broken promise of apple's announced forbidden-photo reporting system – and how to fix it," 8 2021. [Online]. Available: https://www.bu.edu/riscs/2021/08/10/apple-csam/

[31] R. McMillan, J. Stern, and D. Volz, "Apple plans new encryption system to ward off hackers and protect icloud data," *Wall Street Journal*, 12 2022. [Online]. Available: https://www.wsj.com/articles/apple-plans-new-encryption-system-to-ward-off-hackers-and-protect-icloud-data-11670435635

[32] Google, "How hash matching technology helps ncmec," 2021. [Online]. Available: https://safety.google/stories/hash-matching-to-help-ncmec/

[33] Thorn, "Eliminating child sexual abuse material: The role and impact of hash values," 4 2016. [Online]. Available: https://www.thorn.org/blog/eliminating-child-sexual-abuse-material-hash-values/

[34] R. Pfefferkorn, "Content-oblivious trust and safety techniques: Results from a survey of online service providers," *Journal of Online Trust & Safety*, vol. 1, no. 2, 2022.

[35] C. E. for International Peace Encryption Working Group, "Moving the encryption policy conversation forward," 9 2019. [Online]. Available: https://carnegieendowment.org/2019/09/10/moving-encryption-policy-conversation-forward-pub-79573

[36] National Academies of Sciences, Engineering, and Medicine and others, *Decrypting the Encryption Debate: A Framework for Decision Makers*. National Academies Press, 2018.

[37] H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, M. Green, S. Landau, P. G. Neumann *et al.*, "Keys under doormats: mandating insecurity by requiring government access to all data and communications," *Journal of Cybersecurity*, vol. 1, no. 1, pp. 69–79, 2015.

[38] S. Scheffler and J. Mayer, "Sok: Content moderation for end-to-end encryption," 12 2022.

[39] N. C. for Missing and E. Children, "The issues." [Online]. Available: https://www.missingkids.org/theissues

[40] I. Levy and C. Robinson, "Thoughts on child safety on commodity platforms," *arXiv preprint arXiv:2207.09506*, 2022.

[41] K. M. Babchishin, H. L. Merdian, R. M. Bartels, and D. Perkins, "Child sexual exploitation materials offenders: A review." *European Psychologist*, vol. 23, no. 2, p. 130, 2018.

[42] N. M. Malamuth, ""adding fuel to the fire"? does exposure to non-consenting adult or to child pornography increase risk of sexual aggression?" *Aggression and violent behavior*, vol. 41, 2018.

[43] M. C. Seto, R. Karl Hanson, and K. M. Babchishin, "Contact sexual offending by men with online sexual offenses," *Sexual Abuse*, vol. 23, no. 1, pp. 124–145, 2011.

[44] J. Wolak, D. Finkelhor, and K. J. Mitchell, "Trends in arrests for child pornography production: The third national juvenile online victimization study (njov-3)," *Crimes against Children Research Center*, 2012.

[45] R. Broadhurst, "Child sex abuse images and exploitation materials," in *The human factor of cybercrime*. Routledge, 2019, pp. 310–336.

[46] New York v. Ferber, 458 US 747 (1982).

[47] *Survivors' Survey*, Canadian Centre for Child Protection, 2017. [Online]. Available: https://protectchildren.ca/pdfs/C3P_SurvivorsSurveyExecutiveSummary2017_en.pdf

[48] N. Tyagi, P. Grubbs, J. Len, I. Miers, and T. Ristenpart, "Asymmetric message franking: Content moderation for metadata-private end-to-end encryption," in *CRYPTO 2019*. Springer, 2019. [Online]. Available: https://doi.org/10.1007/978-3-030-26954-8_8

[49] A. Kulshrestha and J. R. Mayer, "Estimating incidental collection in foreign intelligence surveillance: Large-scale multiparty private set intersection with union and sum," in *USENIX Security 2022*, 2022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/kulshrestha

[50] *2020 Reports by Electronic Service Providers (ESP)*, National Center for Missing and Exploited Children. [Online]. Available: https://www.missingkids.org/content/dam/missingkids/gethelp/2020-reports-by-esp.pdf

[51] S. Hicks, "Dragonflai," 11 2021. [Online]. Available: https://www.safetytechnetwork.org.uk/provider/dargonflai/

[52] R. Jackson, "T3k.ai," 1 2022. [Online]. Available: https://www.safetytechnetwork.org.uk/provider/t3k-ai/

[53] Galaxkey, "Galaxkey wins uk government funding to enable children to communicate safely online," 11 2021. [Online]. Available: https://www.galaxkey.com/blog/galaxkey-wins-uk-government-funding-to-enable-children-to-communicate-safely-online/

[54] T. Claburn, "Apple didn't engage with the infosec world on csam scanning – so get used to a slow drip feed of revelations," *The Register*, 8 2021. [Online]. Available: https://www.theregister.com/2021/08/18/apples_csam_hashing/

[55] J. Mayer and A. Kulshrestha, "We built a system like apple's to flag child sexual abuse material — and concluded the tech was dangerous," *Washington Post*, 8 2021. [Online]. Available: https://www.washingtonpost.com/opinions/2021/08/19/apple-csam-abuse-encryption-security-privacy-dangerous/

[56] E. Rescorla, "More on apple's client-side csam scanning," *Educated Guesswork*, 8 2021. [Online]. Available: https://educatedguesswork.org/posts/apple-csam-more/

[57] B. Mitchell, K. Kaul, G. S. McNamara, M. Tucker, J. Hicks, C. Bliss, R. Ober, D. Castro, A. Wells, C. Reguerin, C. Green-Ortiz, and K. Stavinoha, *Going dark: impact to intelligence and law enforcement and threat mitigation*, 2017. [Online]. Available: https://www.dni.gov/files/PE/Documents/10---2017-AEP_Going-Dark.pdf

[58] J. Nicas, R. Zhong, and D. Wakabayashi, "Inside apple's compromises in china: A times investigation," *New York Times*, May 2021. [Online]. Available: https://www.nytimes.com/2021/05/17/technology/apple-china-censorship-data.html

[59] M. Isaac, "Whatsapp sues india's government to stop new internet rules," *New York Times*, May 2021. [Online]. Available: https://www.nytimes.com/2021/05/25/technology/whatsapp-india-lawsuit.html

[60] A. Troianovski and A. Satariano, "Apple and google remove 'navalny' voting app in russia," *New York Times*, 9 2021. [Online]. Available: https://www.nytimes.com/2021/09/17/world/europe/russia-navalny-app-election.html

[61] E. Volokh, "The mechanisms of the slippery slope," *Harvard Law Review*, vol. 116, no. 4, pp. 1026–1137, 2003.

[62] E. Douek, "More content moderation is not always better," *Wired*, 6 2021. [Online]. Available: https://www.wired.com/story/more-content-moderation-not-always-better/

[63] S. Jain, A.-M. Cretu, and Y.-A. de Montjoye, "Adversarial detection avoidance attacks: Evaluating the robustness of perceptual hashing-based client-side scanning," *arXiv preprint arXiv:2106.09820*, 2021.

[64] Apple, Inc., *Security Threat Model Review of Apple's Child Safety Features*, 2021. [Online]. Available: https://www.apple.com/child-safety/pdf/Security_Threat_Model_Review_of_Apple_Child_Safety_Features.pdf

[65] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," *IEEE Security & Privacy*, 2016.

[66] B. Dwyer, "Imagenet contains naturally occurring neuralhash collisions," 8 2021. [Online]. Available: https://blog.roboflow.com/neuralhash-collision/

[67] A. Athalye and M. A. KERİMOĞLU, "Neuralhash collider," 9 2021. [Online]. Available: https://github.com/anishathalye/neural-hash-collider

[68] D. Hintersdorf, L. Struppek, D. Neider, and K. Kersting, "Learning to break deep perceptual hashing: The use case neuralhash," *IEEE Workshop on Technology and Consumer Protection*, 2022.

[69] W. Liu, W. Wang, H. Chen, X. Wang, Y. Lu, K. Chen, X. Wang, Q. Shen, Y. Chen, and H. Tang, "Practical and efficient in-enclave verification of privacy compliance," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021.

[70] R. Sinha, S. Rajamani, S. Seshia, and K. Vaswani, "Moat: Verifying confidentiality of enclave programs," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[71] Y. Wang, Y. Shen, C. Su, K. Cheng, Y. Yang, A. Faree, and Y. Liu, "Cfhider: Control flow obfuscation with intel sgx," in *IEEE Conference on Computer Communications*, 2019.

[72] A. Awad, Y. Wang, D. Shands, and Y. Solihin, "Obfusmem: A low-overhead access obfuscation for trusted memories," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017.

[73] J. F. Almansa, I. Damgård, and J. B. Nielsen, "Simplified threshold RSA with adaptive and proactive security," in *EUROCRYPT*, 2006. [Online]. Available: https://doi.org/10.1007/11761679_35

[74] R. Cohen, "Asynchronous secure multiparty computation in constant time," in *PKC 2016*. Springer, 2016. [Online]. Available: https://doi.org/10.1007/978-3-662-49387-8_8

[75] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *International Workshop on Public Key Cryptography*. Springer, 2003, pp. 31–46.

[76] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold dss signatures," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1996, pp. 354–371.

[77] R. Pagh and F. F. Rodler, "Cuckoo hashing," *Journal of Algorithms*, vol. 51, no. 2, pp. 122–144, 2004.

[78] A. Kirsch, M. Mitzenmacher, and U. Wieder, "More robust hashing: Cuckoo hashing with a stash," *SIAM Journal on Computing*, vol. 39, no. 4, pp. 1543–1561, 2010.

[79] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than bloom," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, 2014, pp. 75–88.

[80] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in *USENIX Security*, 2015, pp. 515–530.

[81] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder, "Efficient circuit-based psi via cuckoo hashing," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 125–157.

[82] D. Demmler, P. Rindal, M. Rosulek, and N. Trieu, "Pir-psi: Scaling private contact discovery." *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 4, pp. 159–178, 2018.

[83] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai, "Efficient circuit-based psi with linear communication," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 122–153.

[84] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Psi from paxos: fast, malicious private set intersection," *EUROCRYPT*, 2020.

[85] M. K. Mıhçak and R. Venkatesan, "New iterative geometric methods for robust perceptual image hashing," in *ACM Workshop on Digital Rights Management*. Springer, 2001, pp. 13–21.

[86] B. Yang, F. Gu, and X. Niu, "Block mean value based image perceptual hashing," in *2006 International Conference on Intelligent Information Hiding and Multimedia*. IEEE, 2006, pp. 167–172.

[87] V. Monga and B. L. Evans, "Perceptual image hashing via feature points: performance evaluation and tradeoffs," *IEEE transactions on Image Processing*, vol. 15, no. 11, pp. 3452–3465, 2006.

[88] E. Klinger and D. Starkweather, "pHash: The open source perceptual hash library," 2013. [Online]. Available: https://www.phash.org/

[89] A. Neelima and K. M. Singh, "Perceptual hash function based on scale-invariant feature transform and singular value decomposition," *The Computer Journal*, vol. 59, no. 9, pp. 1275–1281, 2016.

[90] C. B. Project, "Testing different image hash functions," 2 2018. [Online]. Available: https://content-blockchain.org/research/testing-different-image-hash-functions/

[91] J. Prokos, T. M. Jois, N. Fendley, R. Schuster, M. Green, E. Tromer, and Y. Cao, "Squint hard enough: Evaluating perceptual hashing with machine learning," *Cryptology ePrint Archive*, 2021.

[92] Apple, Inc., *CSAM Detection Technical Summary*, 2021. [Online]. Available: https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf

[93] J. Knockel and L. Ruan, "Engrave danger: An analysis of apple engraving censorship across six regions," *The Citizen Lab*, 8 2021. [Online]. Available: https://citizenlab.ca/2021/08/engrave-danger-an-analysis-of-apple-engraving-censorship-across-six-regions/

[94] R. Canetti, A. Cohen, and Y. Lindell, "A simpler variant of universally composable security for standard multiparty computation," *IACR Cryptol. ePrint Arch.*, p. 553, 2014. [Online]. Available: http://eprint.iacr.org/2014/553

[95] Apple, Inc., *Expanded Protections for Children: Frequently Asked Questions*, August 2021. [Online]. Available: https://www.apple.com/child-safety/pdf/Expanded_Protections_for_Children_Frequently_Asked_Questions.pdf

[96] T. Cook, "A message to our customers," 2 2016. [Online]. Available: https://www.apple.com/customer-letter/

[97] Apple, "iCloud operated by GCBD Terms and Conditions," 11 2021. [Online]. Available: https://www.apple.com/legal/internet-services/icloud/en/gcbd-terms.html

[98] B. Libert and M. Yung, "Concise mercurial vector commitments and independent zero-knowledge sets with short proofs," in *Theory of Cryptography TCC 2010*. Springer, 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-11799-2_30

[99] D. Catalano and D. Fiore, "Vector commitments and their applications," in *PKC 2013*. Springer, 2013. [Online]. Available: https://doi.org/10.1007/978-3-642-36362-7_5

[100] J. C. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital sinatures (extended abstract)," in *EUROCRYPT '93*. Springer, 1993. [Online]. Available: https://doi.org/10.1007/3-540-48285-7_24

[101] D. Boneh, B. Bünz, and B. Fisch, "Batching techniques for accumulators with applications to iops and stateless blockchains," in *CRYPTO 2019*. Springer, 2019. [Online]. Available: https://doi.org/10.1007/978-3-030-26948-7_20

[102] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *CRYPTO 2002*. Springer, 2002. [Online]. Available: https://doi.org/10.1007/3-540-45708-9_5

[103] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *EUROCRYPT 2004*. Springer, 2004. [Online]. Available: https://doi.org/10.1007/978-3-540-24676-3_1

[104] A. C. D. Resende and D. F. Aranha, "Faster unbalanced private set intersection," in *Financial Cryptography and Data Security 2018*. Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-3-662-58387-6_11

[105] F. Baldimtsi, J. Camenisch, M. Dubovitskaya, A. Lysyanskaya, L. Reyzin, K. Samelin, and S. Yakoubov, "Accumulators with applications to anonymity-preserving revocation," in *IEEE European Symposium on Security and Privacy, EuroS&P*, 2017. [Online]. Available: https://doi.org/10.1109/EuroSP.2017.13

[106] F. Baldimtsi, R. Canetti, and S. Yakoubov, "Universally composable accumulators," in *Topics in Cryptology - CT-RSA*. Springer, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-40186-3_27

[107] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Annual international cryptology conference*. Springer, 1991, pp. 129–140.

[108] J. Knockel and R. Xiong, "(can't) picture this 2," 7 2019. [Online]. Available: https://citizenlab.ca/2019/07/cant-picture-this-2-an-analysis-of-wechats-realtime-image-filtering-in-chats/

[109] E. Cresci, "Russia bans picture of vladimir putin in drag," *The Guardian*, 4 2017. [Online]. Available: https://www.theguardian.com/world/2017/apr/06/russia-bans-picture-of-vladimir-putin-in-drag

[110] X. Wang, A. J. Malozemoff, and J. Katz, "EMP-toolkit: Efficient MultiParty computation toolkit," https://github.com/emp-toolkit, 2016.

[111] X. Wang, S. Ranellucci, and J. Katz, "Authenticated garbling and efficient maliciously secure two-party computation," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 21–37.

[112] D. Archer, V. A. Abril, S. Lu, P. Maene, N. Mertens, D. Sijacic, and N. Smart, "'bristol fashion' mpc circuits," 2020. [Online]. Available: https://github.com/mkskeller/bristol-fashion/blob/master/AES-non-expanded.txt

[113] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," IEEE, 2005, dec 14, 2005 version. [Online]. Available: http://eccc.uni-trier.de/eccc-reports/2001/TR01-016

[114] D. Hofheinz, J. Müller-Quade, and D. Unruh, "On the (im)possibility of extending coin toss," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006, pp. 504–521.

[115] D. Boneh and V. Shoup, "A graduate course in applied cryptography," 2020. [Online]. Available: https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_5.pdf